PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G06F 9/46, 9/44

A1

(11) International Publication Number: **WO 98/49618**

(43) International Publication Date: 5 November 1998 (05.11.98)

(21) International Application Number: PCT/US98/08725

(22) International Filing Date: 30 April 1998 (30.04.98)

(30) Priority Data:
08/846,756 30 April 1997 (30.04.97) US

(71) Applicant: THE FOXBORO COMPANY [US/US]; 33 Commercial Street, Foxboro, MA 01463 (US).

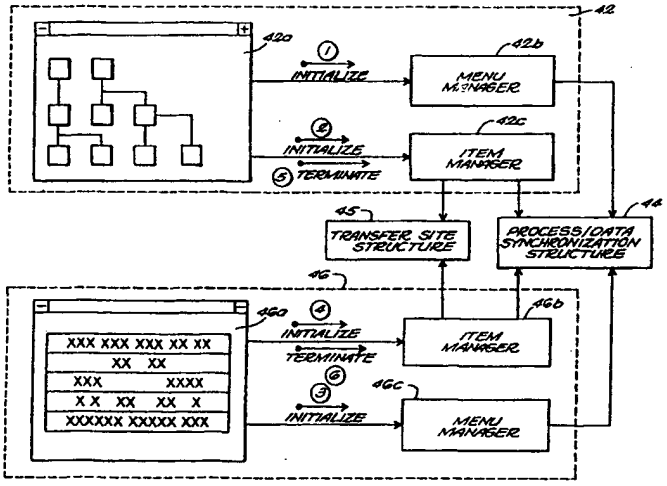
(72) Inventors: ELDRIDGE, Keith, E.; 62 N. Precinct Street, E. Taunton, MA 02718 (US). BUSHY, Dennis, M.; 34 Hunters Run, Franklin, MA 02038 (US).

(74) Agent: POWSNER, David, J.; Choate, Hall & Stewart, Exchange Place, 53 State Street, Boston, MA 02109 (US).

(81) Designated States: AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GE, GH, GM, GW, HU, ID, IL, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, UZ, VN, YU, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, NE, SN, TD, TG).

Published
With international search report.

(54) Title: METHODS AND SYSTEMS FOR SYNCHRONIZING PROCESSES EXECUTING ON A DIGITAL DATA PROCESSING SYSTEM



(57) Abstract

A method and system for synchronizing plural processes executing on a digital data processing system include the steps of registering each of the processes for notification of at least selected events occurring in the other processes. Those events can include, for example, the addition, deletion or selection of an item in another process, the selection of the menu item in the graphical user interface of another process, and the invocation or termination of another process. An item is any informational entity in a process, such as the datum or display object. The method and system further detect an event in any of the processes and determining whether that event is one for which the process (other than that in which the event occurred) is registered for notification. If so, that other process is notified of the event, e.g., so that it can take an action based on that effected in connection with the detected event in the process in which it occurred.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	**ES**	Spain	**LS**	Lesotho	**SI**	Slovenia
AM	Armenia	**FI**	Finland	**LT**	Lithuania	**SK**	Slovakia
AT	Austria	**FR**	France	**LU**	Luxembourg	**SN**	Senegal
AU	Australia	**GA**	Gabon	**LV**	Latvia	**SZ**	Swaziland
AZ	Azerbaijan	**GB**	United Kingdom	**MC**	Monaco	**TD**	Chad
BA	Bosnia and Herzegovina	**GE**	Georgia	**MD**	Republic of Moldova	**TG**	Togo
BB	Barbados	**GH**	Ghana	**MG**	Madagascar	**TJ**	Tajikistan
BE	Belgium	**GN**	Guinea	**MK**	The former Yugoslav Republic of Macedonia	**TM**	Turkmenistan
BF	Burkina Faso	**GR**	Greece			**TR**	Turkey
BG	Bulgaria	**HU**	Hungary	**ML**	Mali	**TT**	Trinidad and Tobago
BJ	Benin	**IE**	Ireland	**MN**	Mongolia	**UA**	Ukraine
BR	Brazil	**IL**	Israel	**MR**	Mauritania	**UG**	Uganda
BY	Belarus	**IS**	Iceland	**MW**	Malawi	**US**	United States of America
CA	Canada	**IT**	Italy	**MX**	Mexico	**UZ**	Uzbekistan
CF	Central African Republic	**JP**	Japan	**NE**	Niger	**VN**	Viet Nam
CG	Congo	**KE**	Kenya	**NL**	Netherlands	**YU**	Yugoslavia
CH	Switzerland	**KG**	Kyrgyzstan	**NO**	Norway	**ZW**	Zimbabwe
CI	Côte d'Ivoire	**KP**	Democratic People's Republic of Korea	**NZ**	New Zealand		
CM	Cameroon			**PL**	Poland		
CN	China	**KR**	Republic of Korea	**PT**	Portugal		
CU	Cuba	**KZ**	Kazakstan	**RO**	Romania		
CZ	Czech Republic	**LC**	Saint Lucia	**RU**	Russian Federation		
DE	Germany	**LI**	Liechtenstein	**SD**	Sudan		
DK	Denmark	**LK**	Sri Lanka	**SE**	Sweden		
EE	Estonia	**LR**	Liberia	**SG**	Singapore		

Methods and Systems for Synchronizing Processes Executing on a Digital Data Processing System

Background of the Invention

The invention pertains to digital data processing and, more particularly, methods and systems for synchronizing processes executing on a digital data processing system.

Early computer programs were typically written for "stand-alone" operation. Since most computers had limited processor and memory resources, they could execute only a single sequence of instructions at a time. Although databases and other resources were sometimes shared among multiple programs, they were typically executed at different times and, often, by different users. Thus, for example, personnel in the bookkeeping department might use one program to enter records into a computerized accounting log, while the personnel in the accounting department would use another program to print that log. As early user interfaces were relatively unrefined, it would not be unusual for those programs to refer to the accounting log and its data in very different ways.

As processor resources increased and interface techniques improved, programmers began writing software packages that operated simultaneously with one another and that displayed and printed shared data on more uniform bases. Developers of these early integrated packages attempted to insure that data files written by one program in a package could be read by another program in that package. Thus, for example, most packages permitted a table generated by a spreadsheet application to be read into the corresponding word processing application. In addition, many packages capitalized on object embedding technologies to "dynamically" link the data files generated by one application program into those of another. For example, a table generated by a spreadsheet application could be embedded in a word processing document so that changes made by the former were automatically included in the later.

A problem with prior art integrated packages is that they are not sufficiently integrated. For example, although data generated by one application can be dynamically linked into another, both applications typically cannot modify that data. Instead, current technologies restrict modification to the "source" application, i.e., the application that generated the data in the first instance. In addition, only the source application can format an embedded item. Thus, for example, the arrangement of a table that is dynamically embedded into a word processing document is restricted to the spreadsheet application that created that table.

An object of this invention is to provide improved digital data processing methodologies and, more particularly, improved methods for synchronizing processes executing on a digital data processing system.

A related object of the invention is to provide such methods as are suited for processes that share common data or that maintain corresponding data sets.

Another object upset of the invention use to permit such synchronization regardless of whether the processes are executing on the same computer or across networked computers.

Still another object of the invention is to provide methodologies that can be implemented at minimum cost for use with a broad range of applications program and across a wide spectrum of hardware and software platforms.

Summary of the Invention

The foregoing objects are among those attained on the invention which provides, in one aspect, a method for synchronizing multiple processes executing on a digital data processing system, e.g., a single computer or a plurality of computers that are in communication with one another.

The method includes registering each of the processes for notification of at least selected events occurring in the other processes. Those event types can include, for example, item-specific events (e.g., the addition, deletion or selection of an item in another process), menu selection events (i.e., the selection of the menu entry in another process), and the invocation or termination of another process. As used herein, "item" refers to any entity, such as a datum, used or displayed by a process. It can be, for example, an entry, row, or column in a spreadsheet program, an object displayed in a drawing program, a file opened by an editing program, etc.

The method further includes detecting an event in any of the multiple processes and determining whether that event is one for which processes (other than that in which the event occurred) are registered for notification. If so, those other processes are notified, e.g., so that they can take an action consistent with that caused by the detected event in the process in which it occurred.

By way of example, three applications programs executing simultaneously on a computer can register for notification of menu selections (or "picks") occurring in any of the others. When a user selects a menu options (such as FILE-OPEN) in any of the programs, the method notifies the others of that event. In response, they can interrogate a common database or the operating system to determine the specific action taken (e.g., the specific file opened) and, in turn, can take complimentary actions (e.g., opening related files).

By way of further example, an applications program can register for notification of process invocation or termination, e.g., when other programs are opened or exited. As new processes are started or old ones terminated, the registered process can, for example, alter the menu options it provides. More particularly, if a registered process has a menu with options that permit the user to open or switch to another process, that option can be activated or deactivated depending on whether that other process has already been invoked and is active, or whether it has been terminated and is inactive.

Further aspects of the invention provides methods as described above in which a registry, database, or other store is maintained identifying each process and the events of which it is to be notified.

Still further aspects of the invention provides methods as described above in which events occurring in processes are detected by intercepting selected function or subroutine calls made by those processes. This is accomplished with "shell" or "stub" routines having names like those of the functions/subroutines to be intercepted. Thus, for example, where it is known (e.g., as it is in most windowing environments) that processes typically invoke a specified routine in order to obtain menu selections from the user, an interceptor routine of the same name can be linked (e.g., via dynamic linking libraries, or DLL's) prior to linking of the specified routine. Once that interceptor routine is invoked, it can determine the type event and notify the other registered processes accordingly. Thereafter, the interceptor routine can itself invoke the specified routine, thereby, enabling the process in which the event occurred to continue processing in the normal course.

Other aspects of the invention provides methods as described above including the step of detecting data transfer events, such as "drag-and-drop" events, between any of the processes. Drag-and-drop events are events in which a user "drags" an item from the window of a first process to the window of a second process, thereby, causing that information to be added to the second process. Upon detection of a drag-and-drop event, the method determines whether the recipient of that operation is registered to receive data in that manner and, if so, at what site in its "window" and in what format. If the operation proceeds, the method that notifies other processes of the addition of an item to the recipient process. Those other processes can respond, in turn, by adding corresponding items to their own data sets.

Still other aspects of the invention contemplate systems operating in accord with the above-described methodologies for synchronizing multiple processes executing on a digital data processing apparatus.

Yet still other aspects of the invention contemplate articles of manufacture, e.g., magnetic disks, composed of computer usable media embodying a computer program that causes a digital data processing apparatus to synchronize multiple processes executing on such apparatus.

Methods and systems as described above can be advantageously used to synchronize multiple processes executing on a digital data processing system and, if desired, to insure the coherency of data maintained by them. The methods and systems permit all processes to be notified whenever a change is made to one of them so that, for example, as menu selections and item selections are made in one process, they can be mirrored in the other processes. And further, so that data sets independently maintained by each process can be updated for accord with those maintained by the other processes. And still further, so that as items are added to a process, or deleted therefrom, the other processes can update themselves accordingly.

Brief Description of the Drawings

A further understanding of the invention may be attained by reference to the drawings in which:

Figure 1 depicts a digital data processing system of the type in which the invention is practiced;

Figure 2 depicts initialization and termination sequences in a method according to the invention;

Figure 3 depicts a menu selection synchronization sequence in a method according to the invention;

Figure 4 depicts an item selection synchronization sequence in a method according to the invention;

Figure 5 depicts a drag-and-drop synchronization sequence in a method according to the invention;

Figures 6A - 6B illustrate the effect of menu selection synchronization in a method according to the invention;

Figures 7A - 7B illustrate the effect of item selection synchronization in a method according to the invention; and

Figures 8A - 8B illustrate the effect of drag-and-drop synchronization in a method according to the invention;

Figure 9 depicts an article of manufacture embodying a program intended to cause a computer to perform methods according to the invention.

Detailed Description of the Illustrated Embodiment

Operating Environment

Figure 1 depicts a digital data processing system 10 of the type in which the invention is practiced. The illustrated system 10 includes a conventional digital data processor 20 (e.g., a workstation, personal computer, hand-held computing device, etc.) having one or more central processing units 22, main memory 24, input-output system 26, and disk drive (or other mass storage device) 28, all of the conventional type known in the art

Digital data processor 20 is coupled to a monitor 29 for the display of output generated by applications programs, operating systems and other software executing thereon. Other output devices, such as printers and plotters (not shown), may also be

coupled to digital data processor 20. Likewise, input devices, such as a keyboard and "mouse" (not shown), may be attached to the unit 20 to accept user input.

Illustrated digital data processing system 10 further includes a conventional digital data processor 30, which is coupled to digital data processor 20 via network 34. The digital data processor 30 can also include central processing units, main memory, include-output systems, mass storage devices, monitors, output devices and input devices, as illustrated. The network 34 comprises any conventional digital data processing network (e.g., LAN or WAN), cable television-based network, wireless network and/or any telecommunications-based network capable of supporting communications between digital data processors 20, 30.

The drawing shows another conventional digital data processor 32 which is coupled to digital data processors 20, 30, but which does not execute processes that are synchronized by the invention. Rather, digital data processor 32 supplies data 40 that is displayed and/or manipulated by synchronized processes 42, 46, 48 on digital data processors 20, 30. That data 40 can represent, for example, information regarding a process, device, or combination thereof amenable to modeling or control. Data 40 can be, for example, operational information regarding a process control system of the type conventionally used to monitor and control a manufacturing process. Data 40 can also represent, by way of further example, configuration parameters for digital data processing network of the type used in a conventional corporate environment. Data 40 is supplied to processes 42, 46, 48 over network 34 in the conventional manner. A single copy of that data may reside in the stores of digital data processor 32, although, separate copies of that data can be maintained by each of processes 42, 46, 48 in the memories of their associated digital data processors 20, 30. As Figure 1 shows, processes 42, 46, 48 synchronized by the invention can operate on, or share, common data supplied by an external source. However, this is not a requirement of the invention -- which can synchronize processes regardless of their sources of data and regardless of whether that data is shared.

once the interceptor function has been invoked, it notifies the other processes 46 of the menu selection events in process 42 and, particularly, in applications program 42a.

The menu manager 42b also adds, to applications program 42a, menu options for every applications program (i.e., other than program 42a) that will serve as a menu selection for program 42a. Those other applications programs are registered for this purpose by providing, for each of them, a companion file that has the same basic filename as the executable form of the registered program but with a preselected filename extension (e.g., ".mnu"). The menu manager detects those companion files (e.g., by searching for their ".mnu" extensions) and adds corresponding menu options to applications program 42a. In each case, the added menu option consists of the contents of the companion file. If the user selects any of those options at runtime, the menu manager invokes the corresponding executable file.

Thus, for example, if the programs EDITOR and CATALOG will serve as menu selections for program 42a, they are registered by supplying companion files "editor.mnu" and "catalog.mnu". Upon identifying those files, the menu manager 42b adds menu options "editor" and "catalog" to the menu of applications program 42a. If either of those options is selected, a corresponding executable file name is constructed from the option name and the appropriate executable extension, e.g.,"editor.exe" and "catalog.exe" in the case of Windows 95, and that file is executed.

With reference to Figure 2, in step (1), the applications program 42a initializes the menu manager 42b by invoking its InitializeManager entry point. This causes the menu manager 42b to initialize data structures used at runtime, particularly, the process/data synchronization structure 44, which is a common data structure utilized by all processes 42, 46, 48 synchronized by the invention. That structure 44 lists all of the synchronized processes and the types of events of which they are to be notified. Those events can include, for example, menu selection events (i.e., the selection of menu options in a process) and the invocation or termination of another process.

In a preferred embodiment of the invention, the process/data synchronization structure 44 has a structure that is defined as follows:

- process identification
- process type
- for each notification type, the type of notification desired

Upon being initialized by the applications program 42a, the menu manager 42b updates the process/data synchronization structure to register that process for notification of menu selection events occurring in the other synchronized processes, e.g., process 46. At the same time, the menu manager 42b employs the item manager 42c to include an instance of program 42a in the process/data synchronization structure. Other processes can adjust their own menus using the menu manager and that structure's information, e.g., deactivating menu selections that could otherwise be used to initiate program 42a.

With continued reference to Figure 2, item manager 42c works in conjunction with conventional functionality provided in the windowing API of the digital data processor 20 to identify item specific events in the applications program 42a and to notify the other processes 46 of the same.

In an embodiment of the invention for use with the Windows 95 operating system, the item manager 42c operates in connection with that system's standard API notification input routines, i.e., the routines that identify items acted upon by the user in the window of the applications program 42a. Unlike the menu manager, which intercepts calls made by the applications program 42a to the windowing system, the item manager 42c is directly invoked by the applications program as each item-specific event is detected. Thus, for example, the applications program 42a invokes the item manager 42c upon each item selection, deselection, addition, deletion or modification.

With continued reference to Figure 2, in step (2), the applications program 42a initializes the item manager 42b by invoking its InitializeManager entry point. More particularly, the applications program 42a invokes that entry point and specifies whether

the program is to be notified of item-specific events, to wit, the addition, deletion or selection of an item in another process. The applications program 42a also specifies whether that program 42a may receive data via data transfer operations, such as drag-and-drop and/or cut-and-paste operations, and, if so, the permissible format and "drop" site (in the case of drag-and-drop) for such operations. As those skilled in the art will appreciate, a "drag-and-drop" is a common windowing operation wherein data graphically selected in one applications program is"dragged" to another applications program and, thereby, added to that other program. The"drop" site is the location in the recipient applications program to which an icon representing the data is dragged. A "cut-and-paste" is a common windowing operation wherein data selected in one applications program is copied (e.g., via a keyboard or mouse command) to a "clipboard" and, subsequently, "pasted" into another applications program.

Item manager 42c stores data transfer information for each process in a common data structure referred to as the transfer site structure 45. In a preferred embodiment of the invention, that structure is defined as follows:

- process identification
- region identification
- ownership indicator
- type of data transfers allowed (e.g., drag-and-drop, cut-and-paste, etc.)

In steps (3)-(4), the applications program 46a of process 46 similarly invokes its respective menu manager 46b and item manager 46c to register the program 46a for notification of desired events, e.g., menu selection events, item-specific events, and drag-and-drop events. As above, that registration information is stored in the common data structures 44 and 45.

The processes 42, 46 can be de-registered from synchronization by transmission of termination notice to their respective item managers 42c, 46c. This causes entries for the respective processes 42, 46 to be removed from the shared structures data structures 44, 45, thus, obviating their notification of further events occurring in the other processes. This is indicated by steps (5) - (6) in the drawing. Upon termination

of an applications program, the item managers can notify all other processes of termination of the process. The notified processes could adjust their menu selections accordingly, e.g., by activating or deactivating menu options to invoke the terminated process.

Synchronization via Menu Selection

Figure 3 depicts a sequence for synchronizing processes 42, 46 based on a menu selection event occurring in one of the processes, to wit, application program 42a. In step (1), the menu manager 42b detects a menu selection event arising from the user's selection of a menu item in the applications program 42a. Such a selection is intercepted by the menu manager 42b in the manner discussed above.

An identification of the menu option chosen by the user is reflected by a value MENU_ID intercepted by the menu manager 42b. In a preferred embodiment of the invention, MENU_ID has a commonly-known value representing the type of menu selection chosen by the user, e.g., FILE-NEW, FILE-OPEN, FILE-CLOSE. Those skilled in the art will, of course, appreciate that MENU_ID may take on other values reflecting that a menu selection event occurred and/or the type of that event.

In step (2), the menu manager 42b passes the MENU_ID to the item manager 42c for possible notification of the other applications programs 46, 48. A preferred menu manager 42b of the invention does not pass MENU_ID's for all intercepted menu selection events to item manager 42c. Rather, it only passes those identifications for menu selections that effect the opening or closing of files, such as FILE-NEW, FILE-OPEN, FILE-CLOSE. Those skilled in the art will, of course, appreciate that MENU_ID's for all or other menu selection events occurring in applications program 42a can be passed to item manager 42c.

In step (3), the item manager 42c determines which processes -- other than that in which the menu selection event was detected -- are registered for notification of menu selection events. This is determined from the process/data synchronization

structure 44, which identifies processes registered for notification of such events, as discussed above.

In step (4), the item manager 42c notifies each of the interested registered processes 46, 48 of the detected menu selection event. Particularly, the item manager transmits the MENU_ID value to each of the registered applications programs 46a, 48a. Such transmission can be accomplished using traditional inter-process communication (IPC) techniques, such as provided by the Windows 95 function SendMessage. Although all of the registered processes can be identified simultaneously, e.g., by a "broadcast" communication, a preferred embodiment of the invention notifies the registered processes serially. Each notified applications program 46a, 48a can respond to the notification, e.g., by opening or closing a corresponding file, creating a corresponding database, etc.

Once the registered processes have been notified, the menu manager 42b permits the process in which the menu selection event was detected (i.e., applications program 42a) to continue. To this end, menu manager 42b invokes the windowing API function that was originally called by that program 42a but that was intercepted by the same-named menu manager interceptor routine.

Synchronization via Item Selection, Addition, Deletion

Figure 4 depicts a sequence for synchronizing processes 42, 46 based on an item-specific event (e.g., an item selection, addition or deletion) occurring in one of the processes, to wit, application program 46a. As noted above, the methods described herein can be extended to the other processes, e.g., process 48, as well.

In step (1), the applications program 42a invokes item manager 42c to register the program 42a for notification of specified item-specific events. The illustrated embodiment permits such an applications program to register for item selection, addition or deletion. Other embodiments of the invention can permit registration for notification of other item-specific events, such as highlighting, inactive/active state, etc.

In step (2), the item manager 42c stores in the process/data synchronizing structure 44 an identification of the process 42 (or of the applications program 42a), along with an indication of the types of item-specific events for which it is to be notified. The process/data synchronization structure 44 is defined as described above.

In step (3), the item manager 46c associated with process 46 detects an item-specific event (e.g., an item selection, addition or deletion) in corresponding applications program 46a. As discussed above, such detection is based on interception of routines in the windowing API of the operating system that identify objects selected, added or deleted by the user in the applications program 46a.

An identification of the specific item chosen by the user is reflected by a value ITEM_ID assigned and transmitted by the applications program 46a to the item manager 46c. In a preferred embodiment of the invention, ITEM_ID has a unique value representing the specific item chosen by the user or assigned directly by the applications program 46a. Those skilled in the art will, of course, appreciate that ITEM_ID may be assigned in other ways, preferably, that uniquely identify each item among all of the registered processes.

In step (4), the item manager 46c determines which processes -- other than that in which the item-specific event was detected -- are registered for notification of item-specific events. This is determined from the process/data synchronization structure 44, which (as discussed above) identifies processes registered for notification of such events.

In step (5), the item manager 46c notifies each of the registered processes of the detected item specific event. Particularly, the item manager 46c transmits the ITEM_ID(s) affected by the event to each of the registered applications programs 42a. Such transmission can be accomplished using traditional inter-process communication techniques, as discussed above. Each notified applications program 42a can respond to the notification, e.g., by selecting, adding or deleting the corresponding item in its own display window.

In embodiments where the applications programs maintain respective data stores reflecting, e.g., the content of a common data store, e.g., data store 40 of digital data processor 32 (Figure 1), a notified applications program 42a can respond to notification to update its respective data store.

Once the registered processes have been notified, the item manager 46b permits the process in which the item-specific event was detected (i.e., applications program 46a) to continue.

Synchronization via Data Transfer Operations

Figure 5 depicts a sequence for synchronizing processes 42, 46 based on data transfer events, such as a drag-and-drop events and/or cut-and-paste events, occurring between two of the processes, to wit, applications program 48a and applications program 46a.

At the outset, each applications program that is a potential recipient of a data transfer operation registers with its respective item manager (i) the types of data transfers that program can receive (e.g., drag-and-drop, cut-and-paste, etc.), (ii) the preferred format in which data is to be transferred, and (iii) for programs registered for drag-and-drop, the valid drop sites. This is illustrated in step(0), wherein applications program 46a registers with item manager 46c to receive drag-and-drop transfers.

In steps (1) - (2), the other applications programs, e.g., 42a, utilizes their corresponding item managers, e.g., 42c, to register for notification of item-specific events. This ensures that the program 42a will be notified of item additions resulting from drag-and-drop data transfers received by other programs, e.g., 46a.

In step (3), the item manager 48c associated with the applications program 48a in which a drag-and-drop event is initiated detects the start of that event. Such detection is based on interception of mouse "down" and mouse "move" notifications by that program 48a from the windowing operating system.

In step (4), the item manager 48c confirms the drop site selected by the user for the drag-and-drop event. As discussed above, valid drop sites for each registered process are stored in the transfer site structure 45. Comparing the information in that instruction with the information supplied by the operating system vis-a-vis the identity of the recipient process 46 and the drop site selected by the user, the item manager 48c permits or aborts the drag-and-drop event. In the event that the event is permitted to continue, the item manager 48c formats the transferred data in the format specified for the recipient program 46a in the transfer site structure 45. The formatted data is then transferred using traditional inter-process communication (IPC) techniques (e.g., SendMessage) of the type discussed above. Once that data is received by the recipient, to wit, applications program 46c, it adds the specified items in the format provided.

Once the data transfer is completed, the item manager 46c associated with the recipient applications program 46a notifies the other processes of the additional item. This is accomplished in steps (5) - (7) of Figure 5 in the same manner discussed above in connection with steps (3) - (5) of Figure 4.

Though the illustrated embodiment concerns drag-and-drop events, those skilled in the art will appreciate that it can be readily applied to other types of inter-applications data transfers, for example, cut-and-paste transfers (e.g., where selected item(s) in a first application are "clipped" to the clipboard and, subsequently, copied into a second application).

Synchronization Illustrated

Figures 6 - 9 depict screen dumps illustrating effects of synchronizing processes in accord with the board's discussed above.

Referring to Figures 6a - 6b, there is shown the effects of synchronizing applications program 42a based on a menu selection event occurring in another applications program 46a. More particularly, as shown in Figure 6a, the two applications programs 42a, 46a are executing simultaneously on a digital data

processing system, e.g., in the manner of the processes executing on processor 30 of Figure 1. With continued reference to Figure 6a, the user (not shown) selects a menu option -- particularly, a FILE-OPEN option -- in applications program 46a. In accord with the methodologies discussed above, e.g., in connection with Figure 3, applications program 42a is notified of the selection. As shown in Figure 6b, that applications program 42a responds to the notification by opening a corresponding file and displaying its data.

Referring to Figures 7a - 7b, there is shown the effects of synchronizing applications programs 46a, 50a based on an item selection event occurring in applications program 42a. More particularly, as shown in Figure 7a, the three applications programs 42a, 46a, 50a are executing simultaneously on a digital data processing system. Those programs are also executing with respect to common data, although they display it in differing formats. As shown in Figure 7a, the user (not shown) selects two items 52, 54 displayed by applications program 42a. In accord with the methodologies discussed above, e.g., in connection with Figure 4, applications programs 46a, 50a are notified of the selection. As shown in Figure 7b, those applications programs 46a, 50a respond to the notification by selecting or highlighting corresponding items displayed by them.

Referring to Figures 8a - 8b, there is shown the effects of synchronizing applications program 46a based on a drag-and-drop event occurring between two other processes 52a, 42a. More particularly, as shown in Figure 8a, the three applications programs 42a, 46a, 52a are executing simultaneously on a digital data processing system. As shown in Figure 8a, the user (not shown) drags items 54 displayed by applications program 52a to applications program 42a. In accord with the methodologies discussed above, e.g., in connection with Figure 5, the items are formatted and the drop site confirmed before it is transferred to the recipient applications program 42a. Moreover, the applications program 46a is notified of the event, particularly, of the addition of items 54' single prime, not double prime to applications program 42a. As shown in Figure 8b, that applications program 46 responds to the notification by adding a corresponding items 54' to its display.

Articles of Manufacture

Figure 9 depicts an article of manufacture, to wit, a magnetic diskette, composed of a computer usable media, to wit, a magnetic disk, embodying a computer program that causes digital data processing system 10, or other such digital data processing apparatus, to operate in accord with the methods described above. The diskette is shown in front view and back view. It is of conventional construction and has the computer program stored on the magnetic media therein in a conventional manner readable, e.g., via a read/write head contained in a diskette drive of image analyzer 40. It will be appreciated that diskette is shown by way of example only and that other articles of manufacture comprising computer usable media on which programs intended to cause a computer to execute in accord with the teachings hereof are also embraced by the invention.

Conclusion

Described herein are improved methods, systems and articles of manufacture for digital data processing meeting the objects set forth above. It will be appreciated that the embodiments illustrated and discussed herein are merely examples of the invention and that other embodiments incorporating modifications thereto fall within the scope of the invention, of which we claim:

1. An event-driven method for synchronizing plural processes on a digital data processing system, the method comprising

A. registering each of the plural processes for notification of at least selected events in the other processes;

B. detecting an event in any of the plural processes,

C. determining whether the detected event is one for which a process, other than that in which the event occurred, is registered for notification; and

D. responding to an affirmative determination in step (C) for notifying the other process of the detected event.

2. A method according to claim 1, comprising the step of selectively updating that other process to reflect a change corresponding to that effected by the detected event in the process in which the event occurred.

3. A method according to claim 2, wherein the updating step includes any of the steps of (i) adding within that other process a datum added to the process in which the event occurred; (ii) deleting from that other process a datum deleted from the process in which the event occurred; (iii) selecting within that other process an item selected in the process in which the event occurred; and (iv) modifying a menu within that other process.

4. A method according to claim 3, wherein step (iv) comprises the step of responding to any of process invocation and process termination for any of activating and deactivating a menu option within that other process.

5. A method according to claim 1, wherein step (A) comprises the step of maintaining a registry of processes and events of which they are to be notified.

6. A method according to claim 5, wherein step (A) includes registering at least selected processes for notification of an event including any of (i) addition of a datum to another process, (ii) deletion of a datum from another process, (iii) selection of a datum in another process, (iv) selection of a menu option in another process, (v) invocation of another process, and (vi) termination of another process.

7. A method according to claim 1, wherein the digital data processing system comprises a plurality of interconnected digital data processors and wherein the plural processes are executing on one or more of those digital data processors.

8. A method according to claim 1, wherein step (D) comprises the step of notifying that other process of (i) an identity of the process in which the detected event occurred and (ii) a type of the detected event.

9. A method according to claim 1, wherein step (B) comprises the step of detecting an event in a process by intercepting a call issued by that process with a routine having a name like that which the intercepted call is directed.

10. A method according to claim 9, wherein step (B) comprises the step of generating a call to the routine to which the intercepted call was directed, after notifying any other processes of the detected event.

11. An event-driven method for synchronizing plural processes on a digital data processing system, the method comprising

A. registering at least selected processes for notification of menu selection events in the other processes;

B. detecting a menu selection event in any of the plural processes,

C. identifying processes, other than that in which the menu selection event was detected, registered for notification of menu selection events and notifying them of the detected event.

12. A method according to claim 11, comprising the step

(D) responding to such notification for selectively executing, within each of the identified processes, an operation in accord with that executed in connection with the detected menu selection event in the process in which it was detected.

13. A method according to claim 12, comprising the step of responding to notification of file access-related menu selection event for executing, within each of the identified processes, an operation for any of initiating access, accessing, and terminating access to a file corresponding to that effected by the file access-related menu selection event in the process in which it was detected.

14. A method according to claim 13, comprising the step of responding to notification of a file-open menu selection event for accessing, within each of the identified processes, a file corresponding to that opened by the menu selection event in the process in which it was detected.

15. A method according to claim 13, comprising the step of responding to notification of a file-close menu selection event for terminating access, within each of the identified processes, to a file corresponding to that to which access is terminated by the menu selection event in the process in which it was detected.

16. A method according to claim 12, wherein step (B) comprises the step of detecting a menu selection event in a process by intercepting a call issued by that process with a routine having a name like that which the intercepted call is directed, and wherein the method further comprises the step of invoking, after step (D), the routine to which the intercepted call was directed.

17. A method according to claim 11, wherein step (C) comprises the step of transferring to each of the identified processes a menu selection signal identifying the detected menu selection event.

18. A method according to claim 11, wherein the digital data processing system comprises a plurality of interconnected digital data processors and wherein the plural processes are executing on one or more of those digital data processors.

19. A method according to claim 11, wherein step (C) comprises the step of notifying the processes registered for notification of menu selection events of (i) an identity of the process in which the menu selection event was detected, (ii) a type of the menu selection event.

20. A method according to claim 19, wherein step (C) comprises the step of notifying the processes registered for notification of menu selection events of menu selection events for any of initiating and terminating access to a file.

21. An event-driven method for synchronizing plural processes on a digital data processing system, the method comprising

A. registering at least selected processes for notification of item-specific events in the other processes;

B. detecting an item-specific event in any of the plural processes,

C. identifying processes, other than that in which the item-specific event was detected, registered for notification of item-specific events and notifying them of the detected event.

22. A method according to claim 21, in which step (B) includes the step of detecting as an item-specific event any of a selection of an item, creation of an item, updating a value of an item, and deletion of an item.

23. A method according to claim 22, comprising the step of responding to such notification for taking action on an item within each of the identified processes in accord with that taken on a corresponding item in connection with the detected item-specific event in the process in which it was detected.

24. A method according to claim 23, comprising the step of responding to such notification for any of selecting, adding, updating and deleting an item in any of the identified processes corresponding to an item effected by the detected item-specific event was taken in the process in which it was detected.

25. A method according to claim 21, wherein the digital data processing system comprises a plurality of interconnected digital data processors and wherein the plural processes are executing on one or more of those digital data processors.

26. A method according to claim 21, wherein step (C) comprises the step of notifying the identified processes of (i) an identity of the process in which the item-specific selection event was detected, (ii) a type of the item-specific selection event.

27. A method according to claim 21, comprising the steps of

detecting a data transfer event between any of the processes;

identifying processes, other than a recipient of the data transfer event, registered for notification of item-specific events and notifying them of addition of an item to a process; and

responding to such notification for taking action within each of the identified processes in accord with the addition of an item to the recipient of the data transfer event.

28. A method according to claim 27, wherein the data transfer event includes any of a drag-and-drop event and a copy-and-paste event.

29. A method according to claim 21, comprising the steps of

detecting a drag-and-drop event between any of the processes;

identifying processes, other than a recipient of the drag-and-drop event, registered for notification of item-specific events and notifying them of addition of an item to a process; and

responding to such notification for taking action within each of the identified processes in accord with the addition of an item to the recipient of the drag-and-drop event.

30. A method according to claim 29, comprising the steps of

registering at least selected processes as recipients of drag-and-drop events and identifying valid drop sites within those processes; and

responding to detection of a drag-and-drop event for verifying validity of a drop site for that event in accord with the registration of recipients thereof.

31. A system for event-driven synchronization of plural processes on digital data processing apparatus, the system comprising

A. means for registering each of the plural processes for notification of at least selected events in the other processes;

B. means for detecting an event in any of the plural processes,

C. means for determining whether the detected event is one for which a process, other than that in which the event occurred, is registered for notification; and

D. means for responding to an affirmative such determination for notifying the other process of the detected event.

32. A system according to claim 31, comprising means for selectively updating that other process to reflect a change corresponding to that effected by the detected event in the process in which the event occurred.

33. A system according to claim 31, wherein the means for registering maintains a registry of processes and events of which they are to be notified.

34. A system according to claim 31, wherein the digital data processing apparatus comprises a plurality of interconnected digital data processors and wherein the plural processes are executing on one or more of those digital data processors.

35. A system according to claim 31, wherein the means for responding notifies the other process of (i) an identity of the process in which the detected event occurred and (ii) a type of the detected event.

36. A system according to claim 1, wherein the means for detecting detects an event in a process by intercepting a call issued by that process with a routine having a name like that which the intercepted call is directed.

37. A system for event-driven synchronization of plural processes on a digital data processing apparatus, the system comprising

A. means for registering at least selected processes for notification of menu selection events in the other processes;

B. means for detecting a menu selection event in any of the plural processes,

C. means for identifying processes, other than that in which the menu selection event was detected, registered for notification of menu selection events and notifying them of the detected event.

38. A system according to claim 37, comprising

D. means for responding to such notification for selectively executing, within each of the identified processes, an operation in accord with that executed in connection with the detected menu selection event in the process in which it was detected.

39 A system according to claim 37, wherein the means for identifying transfers to each of the identified processes a menu selection signal identifying the detected menu selection event.

40. A system according to claim 37, wherein the digital data processing apparatus comprises a plurality of interconnected digital data processors and wherein the plural processes are executing on one or more of those digital data processors.

41. A system according to claim 37, wherein the means for identifying notifies the processes registered for notification of menu selection events of (i) an identity of the process in which the menu selection event was detected, (ii) a type of the menu selection event.

42. A system for event-driven synchronization of plural processes on a digital data processing apparatus, the system comprising:

A. means for registering at least selected processes for notification of item-specific events in the other processes;

B. means for detecting an item-specific event in any of the plural processes,

C. means for identifying processes, other than that in which the item-specific event was detected, registered for notification of item-specific events and notifying them of the detected event.

43. A system according to claim 42, in which the means for detecting detects as an item-specific event any of a selection of an item, creation of an item, updating a value of an item, and deletion of an item.

44. A system according to claim 42, comprising means for responding to such notification for taking action on an item within each of the identified processes in accord with that taken on a corresponding item in connection with the detected item-specific event in the process in which it was detected.

45. A system according to claim 42, wherein the digital data processing apparatus comprises a plurality of interconnected digital data processors and wherein the plural processes are executing on one or more of those digital data processors.

46. A system according to claim 42, wherein the means for identifying notifies the identified processes of (i) an identity of the process in which the item-specific selection event was detected, (ii) a type of the item-specific selection event.

47. A system according to claim 42, comprising

means for detecting a data transfer event between any of the processes;

means for identifying processes, other than a recipient of the data transfer event, registered for notification of item-specific events and notifying them of addition of an item to a process; and

means for responding to such notification for taking action within each of the identified processes in accord with the addition of an item to the recipient of the data transfer event.

48. A system according to claim 42, comprising

means for detecting a drag-and-drop event between any of the processes;

means for identifying processes, other than a recipient of the drag-and-drop event, registered for notification of item-specific events and notifying them of addition of an item to a process; and

means for responding to such notification for taking action within each of the identified processes in accord with the addition of an item to the recipient of the drag-and-drop event.

49. An article of manufacture comprising a computer usable medium embodying program code for causing a digital data processor to carry out a method of synchronizing plural processes on a digital data processing system, the method comprising

A. registering each of the plural processes for notification of at least selected events in the other processes;

B. detecting an event in any of the plural processes,

C. determining whether the detected event is one for which a process, other than that in which the event occurred, is registered for notification; and

D. responding to an affirmative determination in step (C) for notifying the other process of the detected event.

50. An article of manufacture comprising a computer usable medium embodying program code for synchronizing plural processes on a digital data processing system, the method comprising

A. registering at least selected processes for notification of menu selection events in the other processes;

B. detecting a menu selection event in any of the plural processes,

C. identifying processes, other than that in which the menu selection event was detected, registered for notification of menu selection events and notifying them of the detected event.

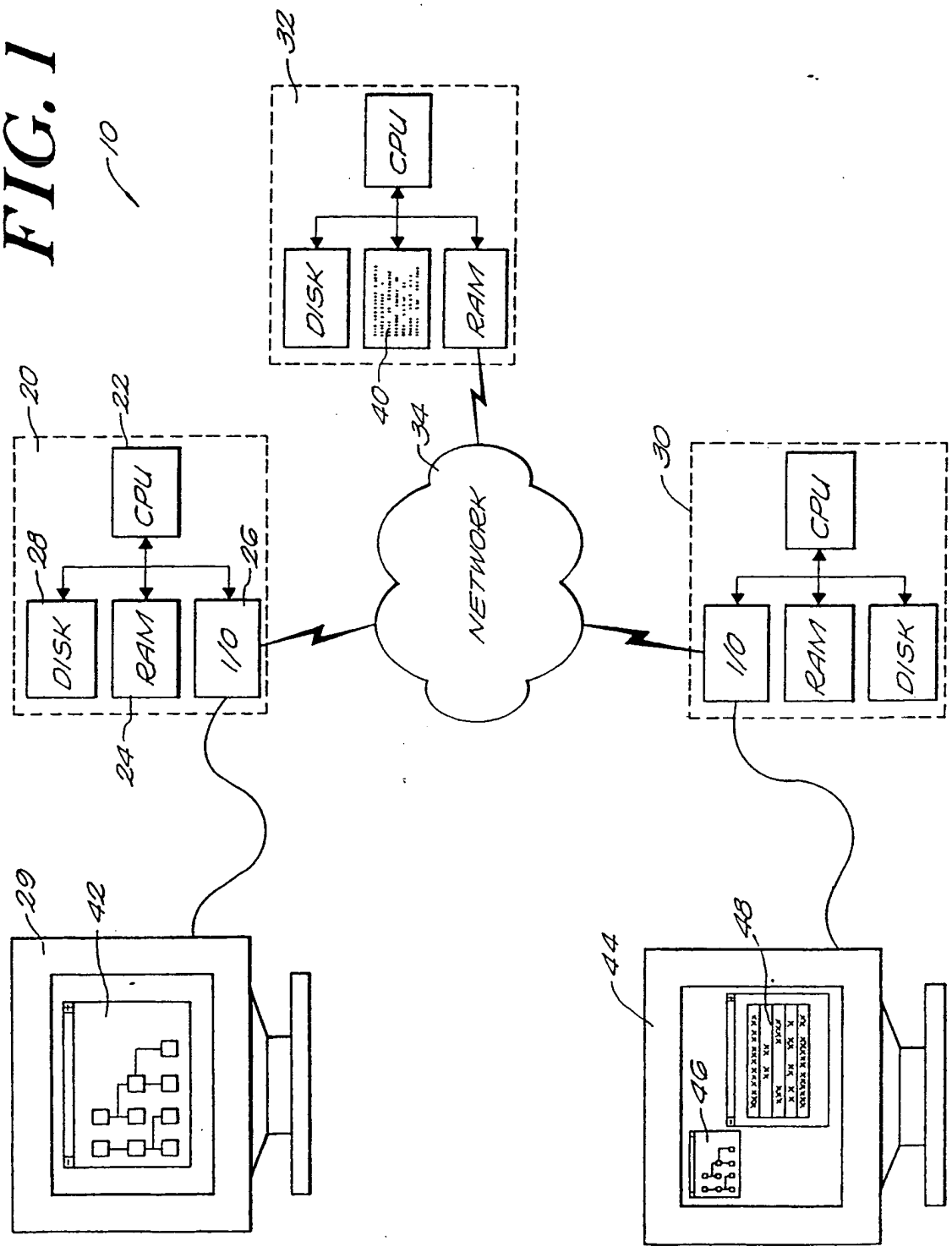
51. An article of manufacture comprising a computer usable medium embodying program code for synchronizing plural processes on a digital data processing system, the method comprising

A. registering at least selected processes for notification of item-specific events in the other processes;

B. detecting an item-specific event in any of the plural processes,

C. identifying processes, other than that in which the item-specific event was detected, registered for notification of item-specific events and notifying them of the detected event.

FIG. 1



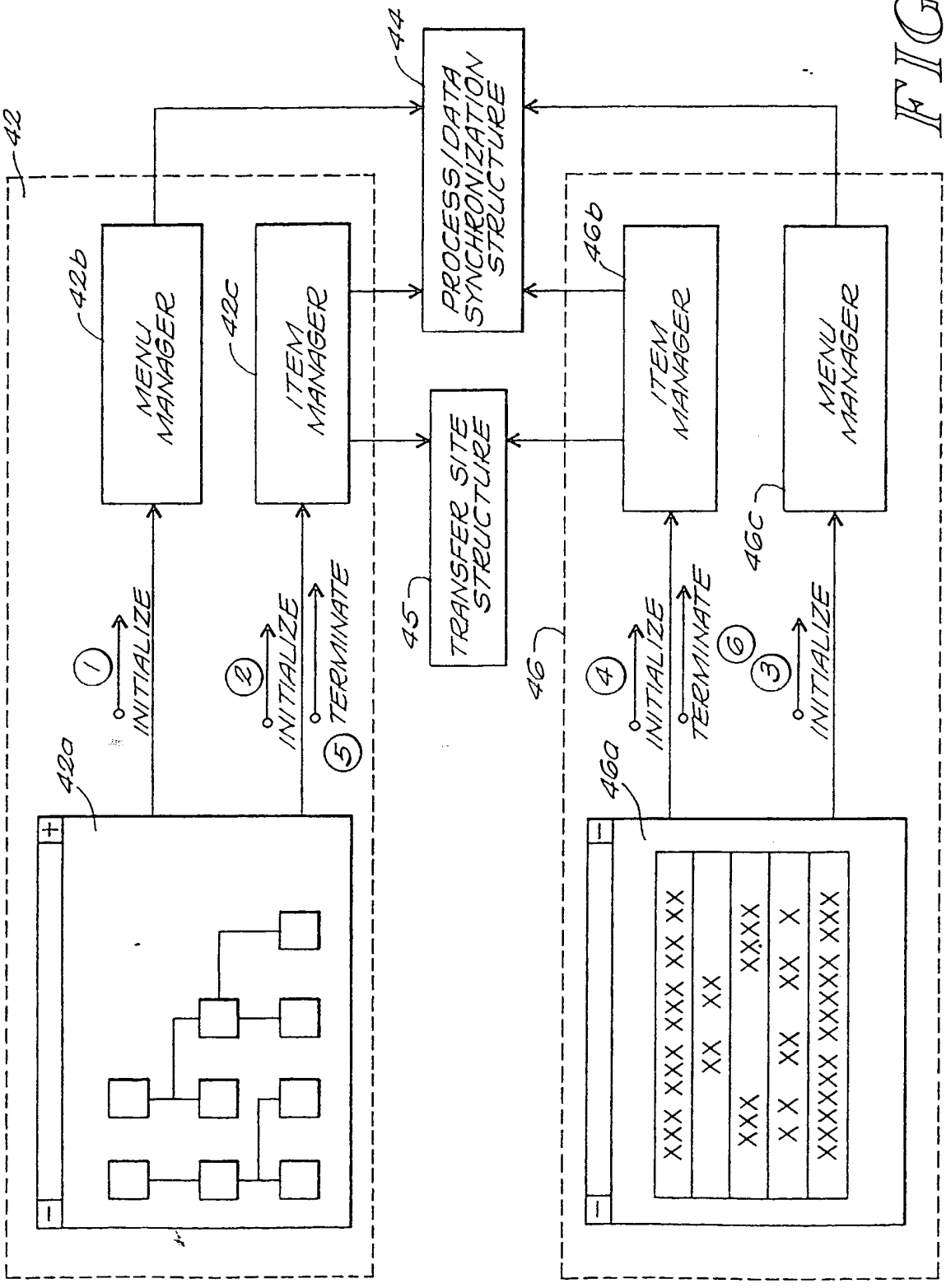


FIG. 2

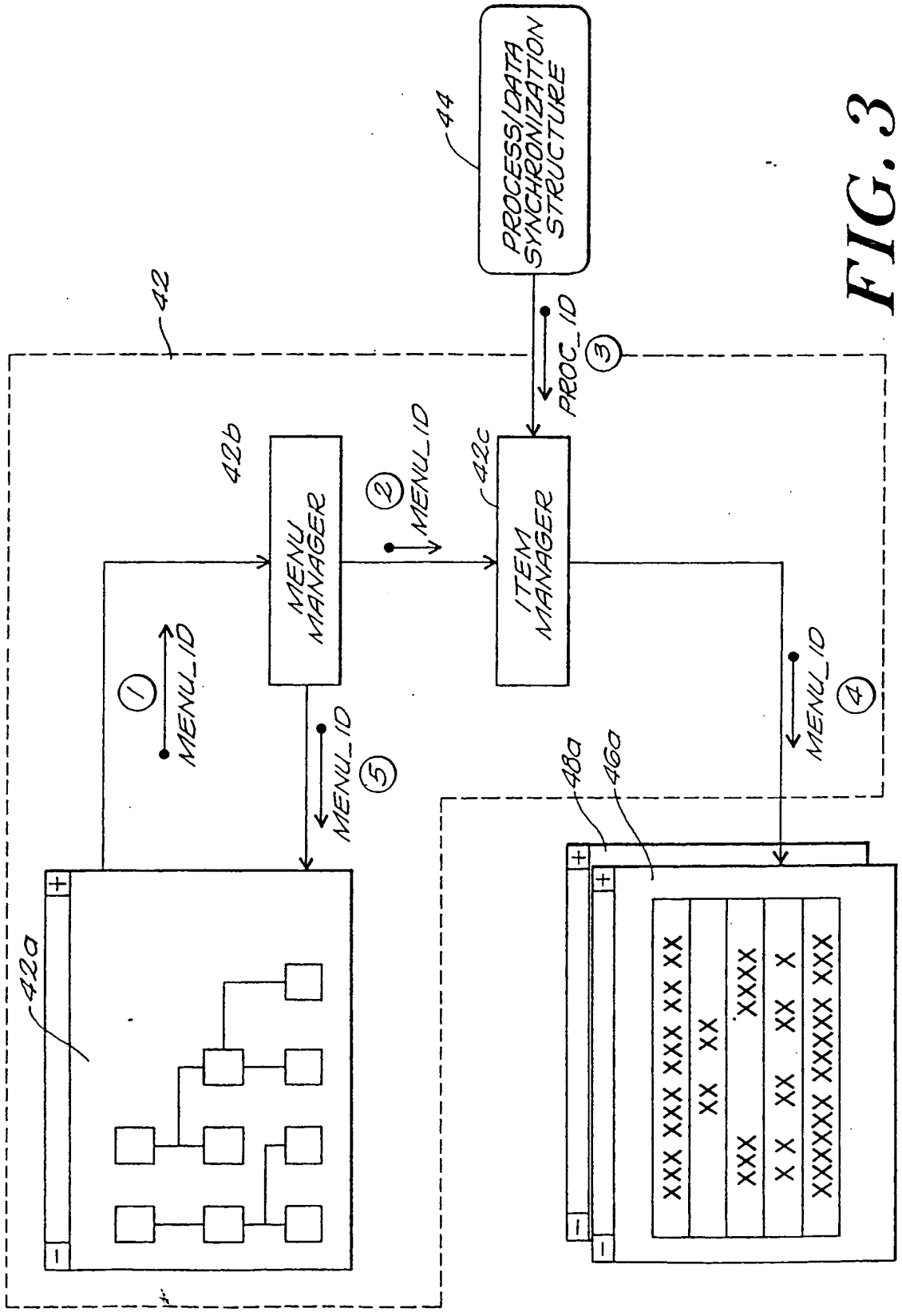


FIG. 3

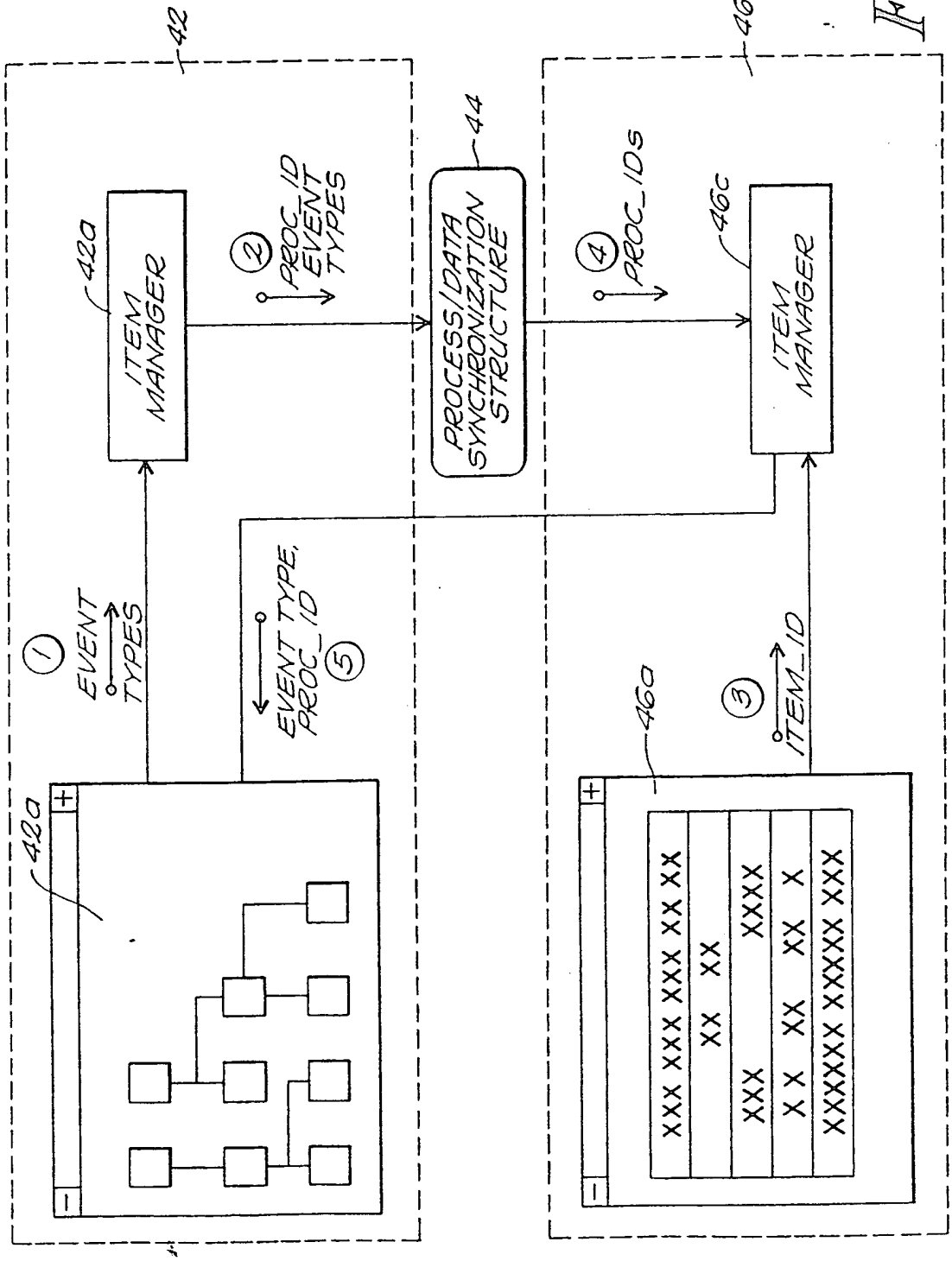


FIG. 4

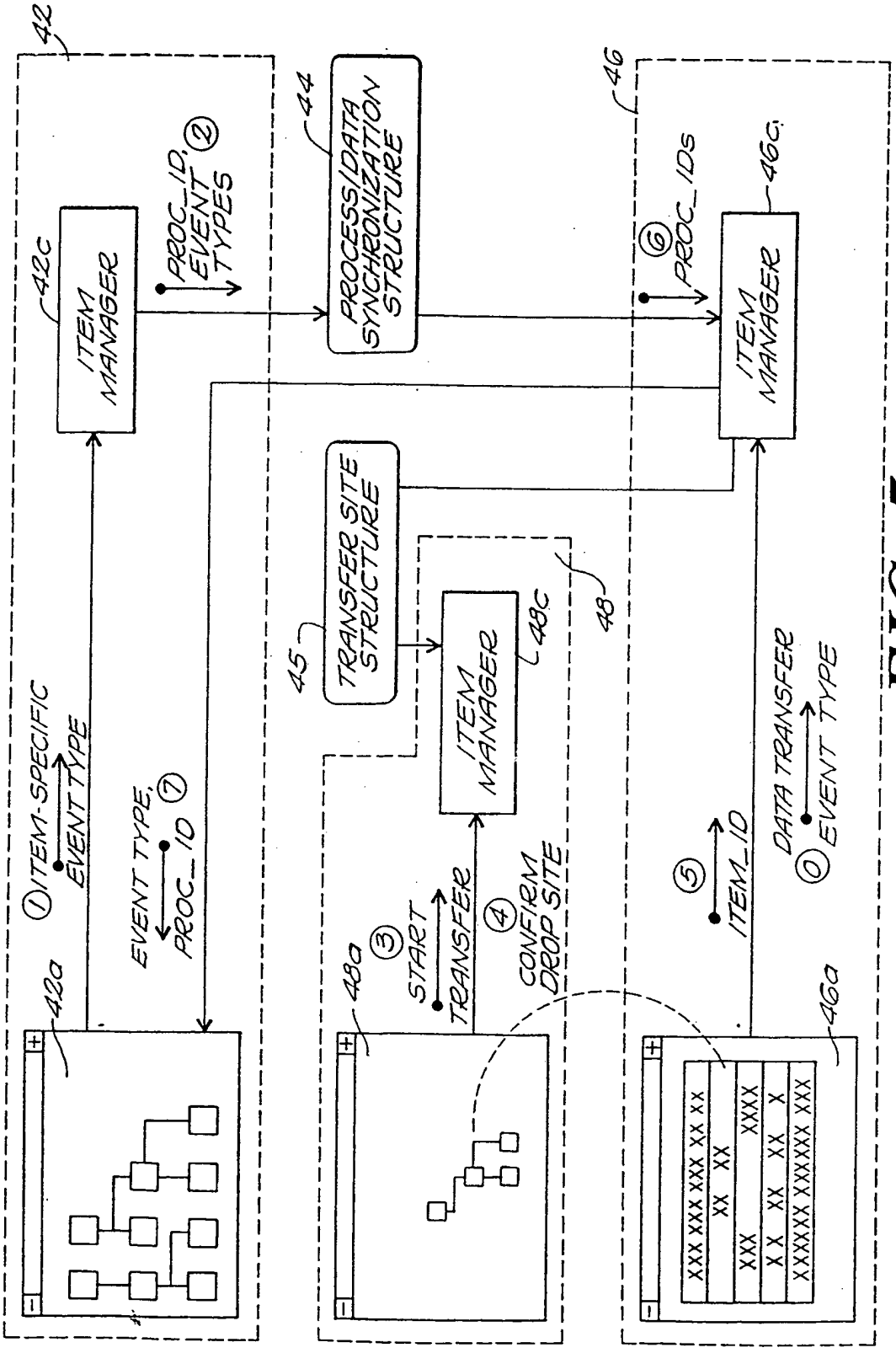


FIG. 5

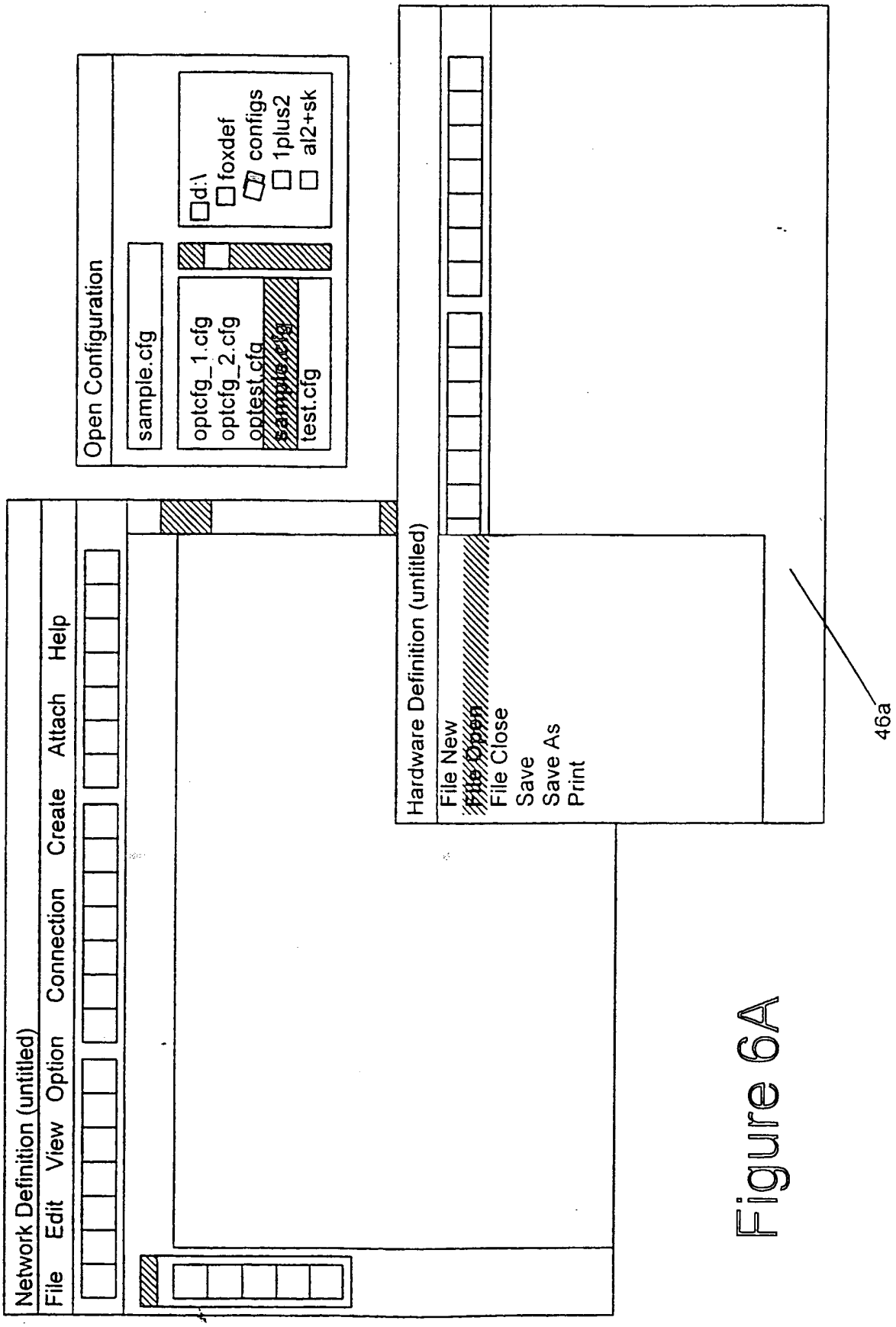


Figure 6A

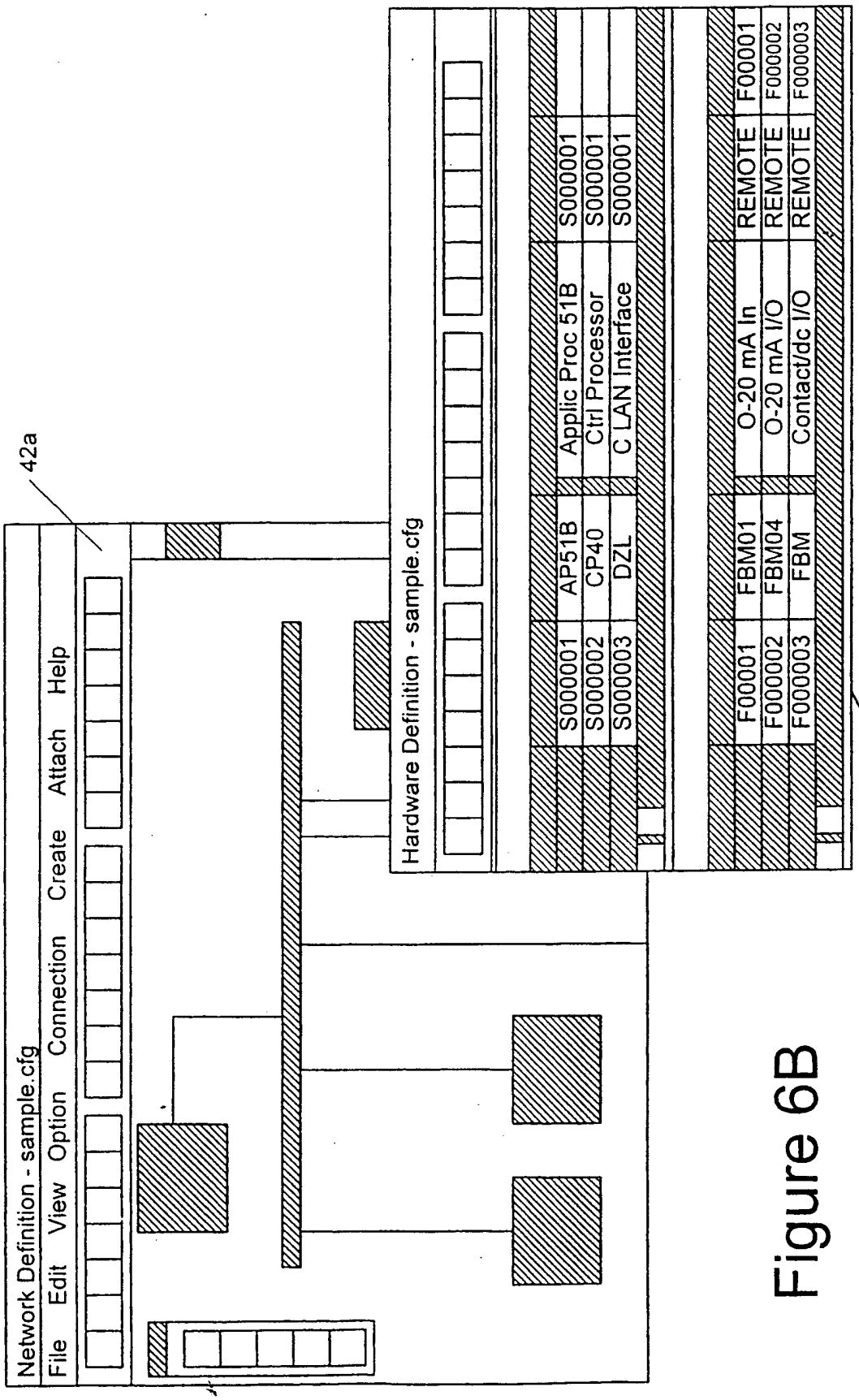

42a
Network Definition - sample.cfg
File Edit View Option Connection Create Attach Help
Hardware Definition - sample.cfg
S000001 AP51B Applic Proc 51B S000001
S000002 CP40 Ctrl Processor S000001
S000003 DZL C LAN Interface S000001
F00001 FBM01 O-20 mA In REMOTE F00001
F000002 FBM04 O-20 mA I/O REMOTE F000002
F000003 FBM Contact/dc I/O REMOTE F000003
46a

Figure 6B

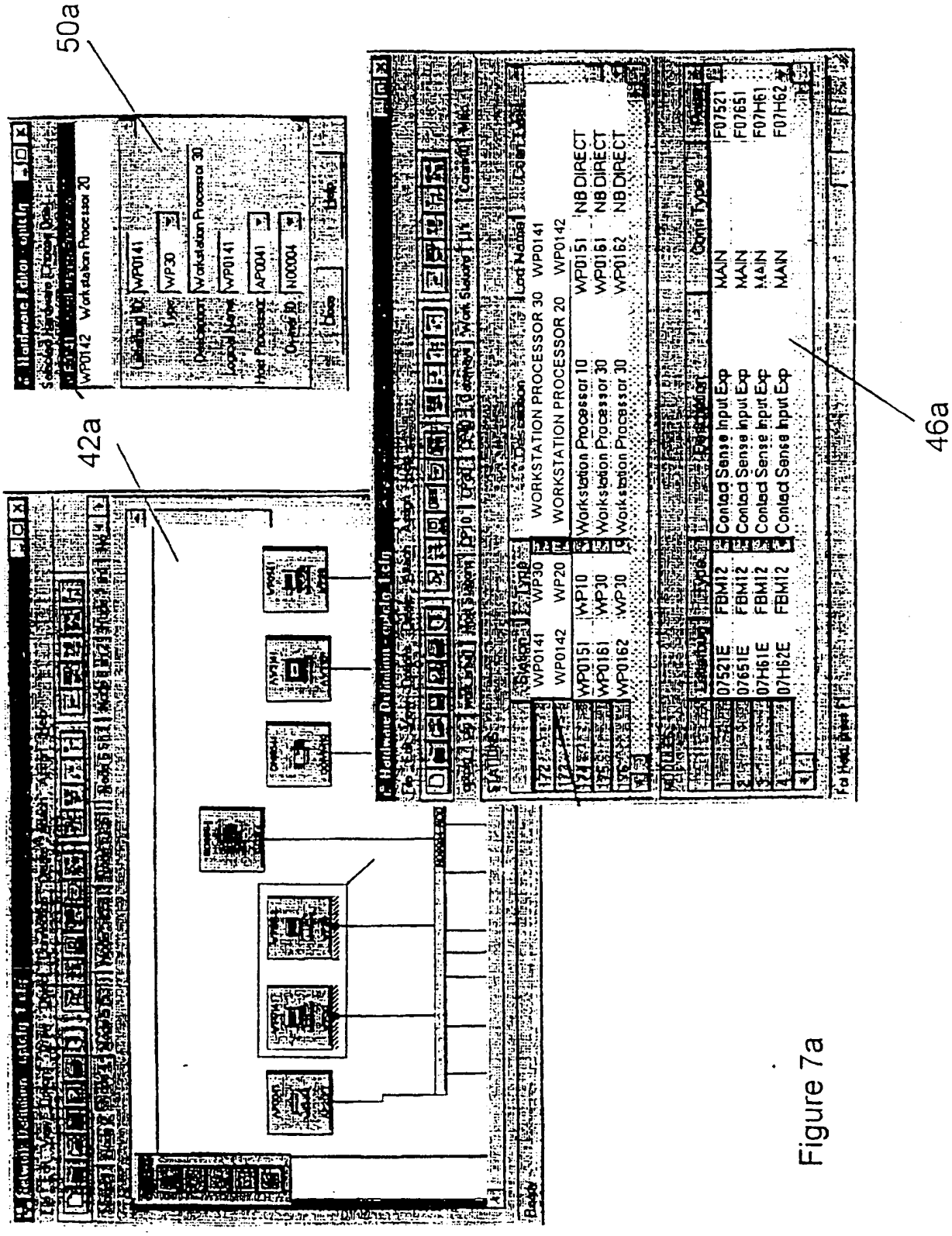


Figure 7a

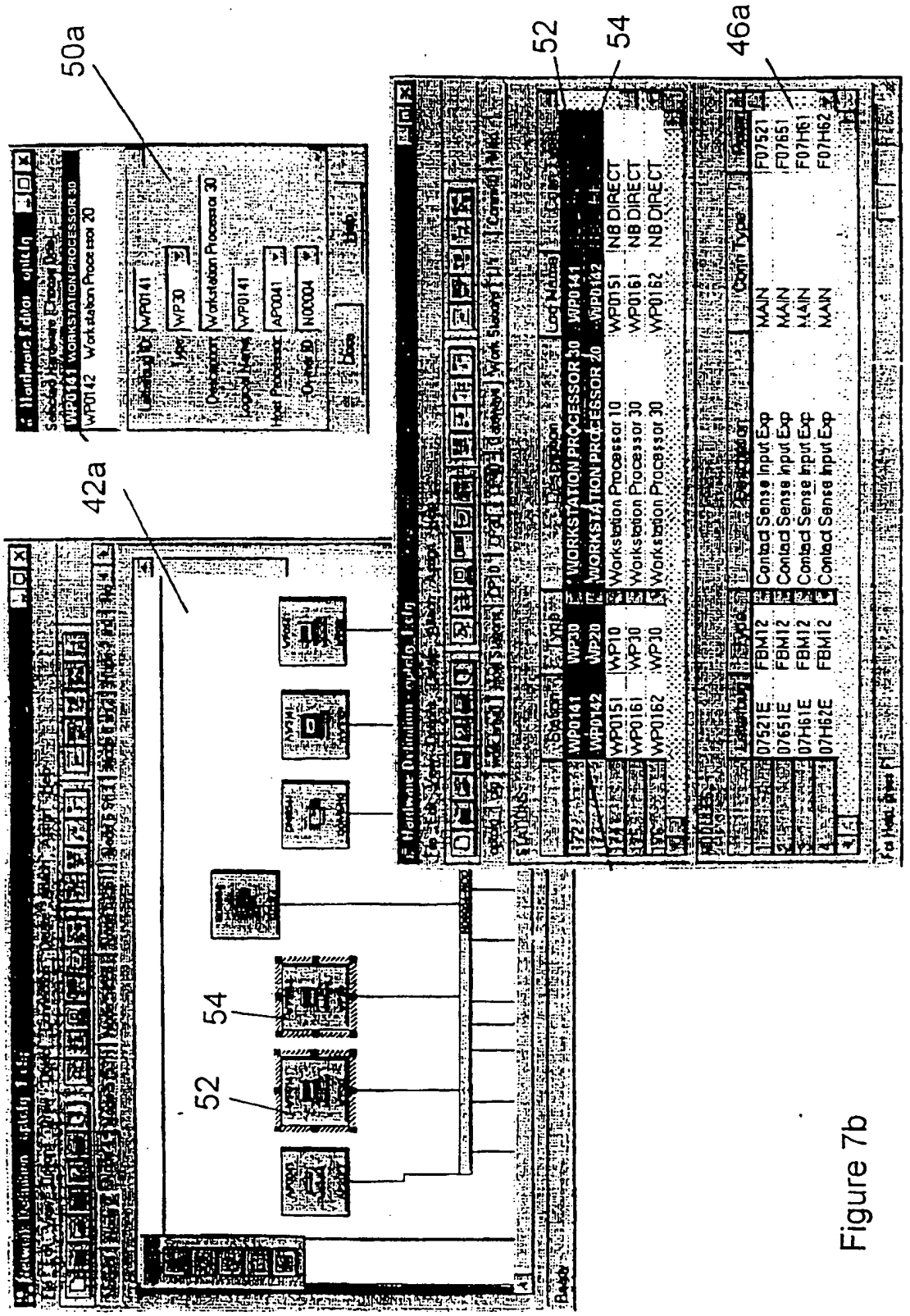


Figure 7b

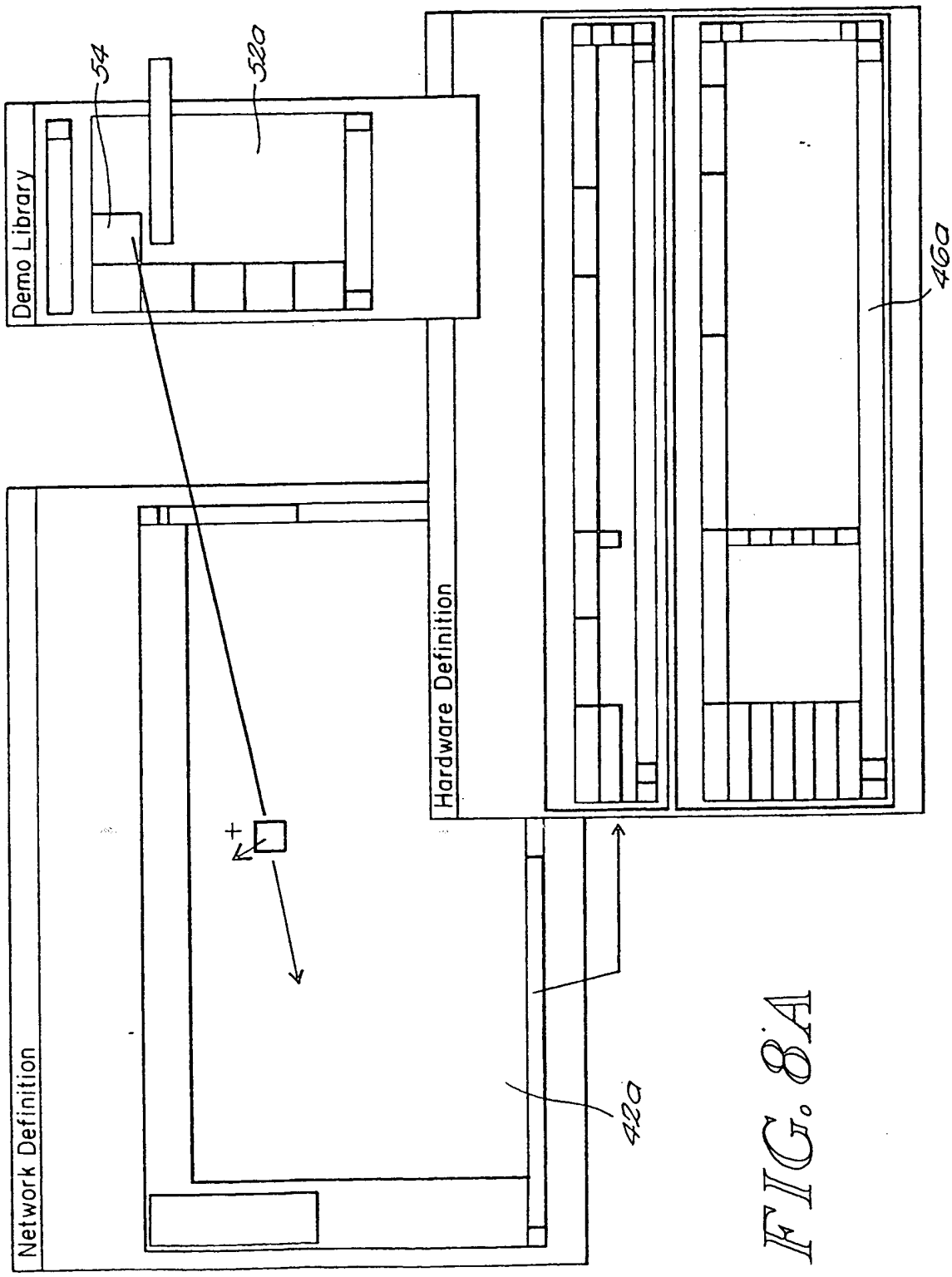


FIG. 8A

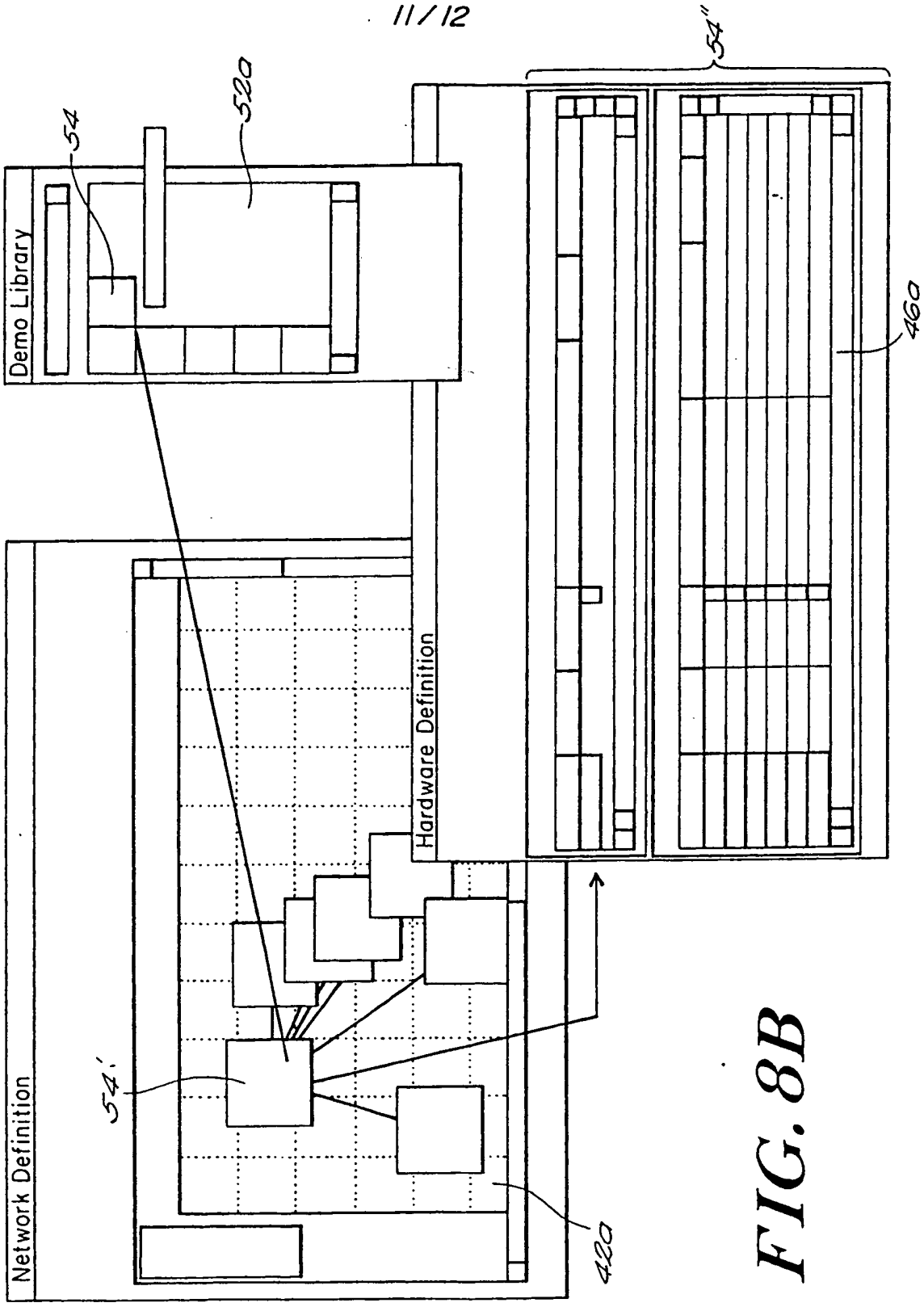

54"
520
54
460
Demo Library
Hardware Definition
Network Definition
54'
420

FIG. 8B

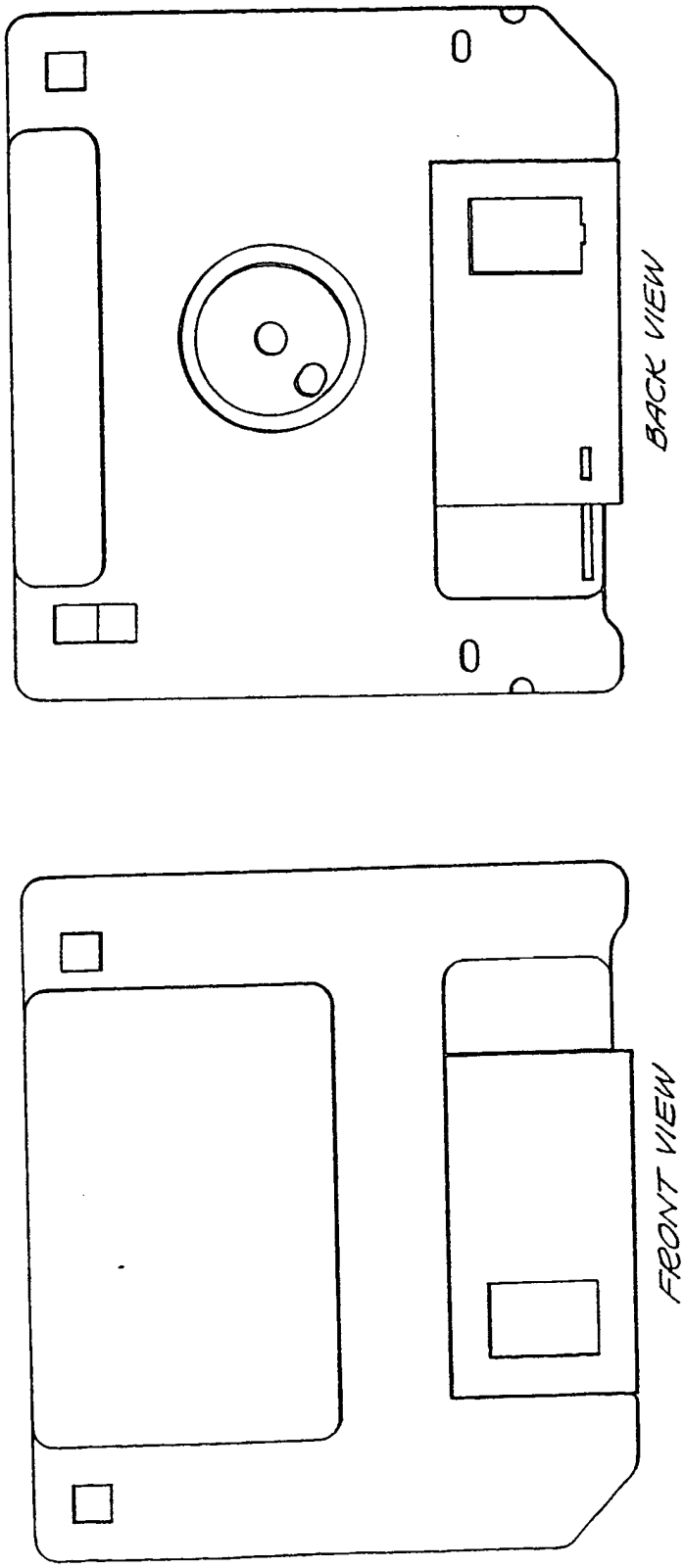


FIG. 9

INTERNATIONAL SEARCH REPORT

National Application No
PCT/US 98/08725

A. CLASSIFICATION OF SUBJECT MATTER
IPC 6 G06F9/46 G06F9/44

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 6 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 5 315 703 A (MATHENY JOHN R ET AL) 24 May 1994 see column 1, line 31 - line 68 see column 9, line 45 - line 57 see column 11, line 29 - column 12, line 37 see column 17, line 15 - column 18, line 40	1-51
X	WO 94 23365 A (KALIEDA LABS INC) 13 October 1994 see page 1, line 8 - page 6, line 5	1-51
A	EP 0 297 339 A (BMC SOFTWARE INC) 4 January 1989 see column 1, line 1 - column 3, line 10	9,10,16, 36

☐ Further documents are listed in the continuation of box C.

☒ Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
4 August 1998	11/08/1998

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Brandt, J

1

Form PCT/ISA/210 (second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

Information on patent family members

National Application No
PCT/US 98/08725

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 5315703 A	24-05-1994	AU 5984594 A	19-07-1994
		CA 2135527 A	07-07-1994
		DE 69310188 D	28-05-1997
		DE 69310188 T	27-11-1997
		EP 0664026 A	26-07-1995
		JP 8501401 T	13-02-1996
		WO 9415285 A	07-07-1994
		US 5367633 A	22-11-1994
		US 5517606 A	14-05-1996
WO 9423365 A	13-10-1994	US 5430875 A	04-07-1995
		AU 674215 B	12-12-1996
		AU 6419194 A	24-10-1994
		CA 2134684 A	01-10-1994
		EP 0643850 A	22-03-1995
		JP 7508116 T	07-09-1995
EP 0297339 A	04-01-1989	AT 134779 T	15-03-1996
		DE 3855029 D	04-04-1996
		DE 3855029 T	05-09-1996
		US 5566334 A	15-10-1996

CORRECTED VERSION*

PCT

WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G06F 9/46, 9/44

A1

(11) International Publication Number: WO 98/49618

(43) International Publication Date: 5 November 1998 (05.11.98)

(21) International Application Number: PCT/US98/08725

(22) International Filing Date: 30 April 1998 (30.04.98)

(30) Priority Data:
08/846,756 30 April 1997 (30.04.97) US

(71) Applicant: THE FOXBORO COMPANY [US/US]; 33 Commercial Street, Foxboro, MA 01463 (US).

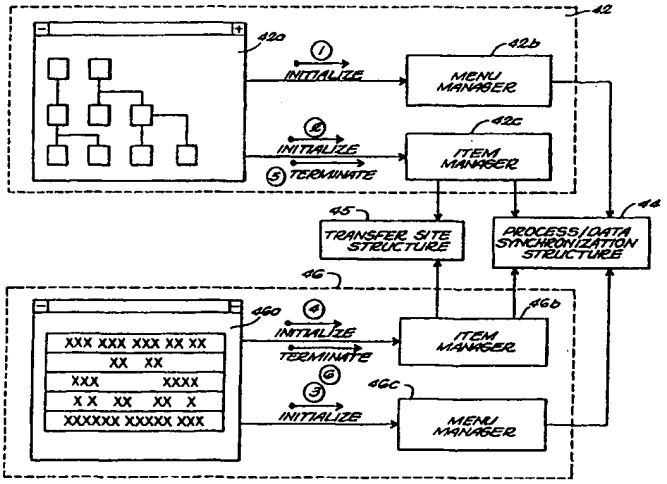
(72) Inventors: ELDRIDGE, Keith, E.; 62 N. Precinct Street, E. Taunton, MA 02718 (US). BUSHY, Dennis, M.; 34 Hunters Run, Franklin, MA 02038 (US).

(74) Agent: POWSNER, David, J.; Choate, Hall & Stewart, Exchange Place, 53 State Street, Boston, MA 02109 (US).

(81) Designated States: AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GE, GH, GM, GW, HU, ID, IL, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, UZ, VN, YU, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, NE, SN, TD, TG).

Published
With international search report.

(54) Title: METHODS AND SYSTEMS FOR SYNCHRONIZING PROCESSES EXECUTING ON A DIGITAL DATA PROCESSING SYSTEM



(57) Abstract

A method and system for synchronizing plural processes executing on a digital data processing system include the steps of registering each of the processes for notification of at least selected events occurring in the other processes. Those events can include, for example, the addition, deletion or selection of an item in another process, the selection of the menu item in the graphical user interface of another process, and the invocation or termination of another process. An item is any informational entity in a process, such as the datum or display object. The method and system further detect an event in any of the processes and determining whether that event is one for which the process (other than that in which the event occurred) is registered for notification. If so, that other process is notified of the event, e.g., so that it can take an action based on that effected in connection with the detected event in the process in which it occurred.

*(Referred to in PCT Gazette No. 13/1999, Section II)

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	**ES**	Spain	**LS**	Lesotho	**SI**	Slovenia
AM	Armenia	**FI**	Finland	**LT**	Lithuania	**SK**	Slovakia
AT	Austria	**FR**	France	**LU**	Luxembourg	**SN**	Senegal
AU	Australia	**GA**	Gabon	**LV**	Latvia	**SZ**	Swaziland
AZ	Azerbaijan	**GB**	United Kingdom	**MC**	Monaco	**TD**	Chad
BA	Bosnia and Herzegovina	**GE**	Georgia	**MD**	Republic of Moldova	**TG**	Togo
BB	Barbados	**GH**	Ghana	**MG**	Madagascar	**TJ**	Tajikistan
BE	Belgium	**GN**	Guinea	**MK**	The former Yugoslav Republic of Macedonia	**TM**	Turkmenistan
BF	Burkina Faso	**GR**	Greece			**TR**	Turkey
BG	Bulgaria	**HU**	Hungary	**ML**	Mali	**TT**	Trinidad and Tobago
BJ	Benin	**IE**	Ireland	**MN**	Mongolia	**UA**	Ukraine
BR	Brazil	**IL**	Israel	**MR**	Mauritania	**UG**	Uganda
BY	Belarus	**IS**	Iceland	**MW**	Malawi	**US**	United States of America
CA	Canada	**IT**	Italy	**MX**	Mexico	**UZ**	Uzbekistan
CF	Central African Republic	**JP**	Japan	**NE**	Niger	**VN**	Viet Nam
CG	Congo	**KE**	Kenya	**NL**	Netherlands	**YU**	Yugoslavia
CH	Switzerland	**KG**	Kyrgyzstan	**NO**	Norway	**ZW**	Zimbabwe
CI	Côte d'Ivoire	**KP**	Democratic People's Republic of Korea	**NZ**	New Zealand		
CM	Cameroon			**PL**	Poland		
CN	China	**KR**	Republic of Korea	**PT**	Portugal		
CU	Cuba	**KZ**	Kazakstan	**RO**	Romania		
CZ	Czech Republic	**LC**	Saint Lucia	**RU**	Russian Federation		
DE	Germany	**LI**	Liechtenstein	**SD**	Sudan		
DK	Denmark	**LK**	Sri Lanka	**SE**	Sweden		
EE	Estonia	**LR**	Liberia	**SG**	Singapore		

Methods and Systems for Synchronizing Processes Executing on a Digital Data Processing System

Background of the Invention

The invention pertains to digital data processing and, more particularly, methods and systems for synchronizing processes executing on a digital data processing system.

Early computer programs were typically written for "stand-alone" operation. Since most computers had limited processor and memory resources, they could execute only a single sequence of instructions at a time. Although databases and other resources were sometimes shared among multiple programs, they were typically executed at different times and, often, by different users. Thus, for example, personnel in the bookkeeping department might use one program to enter records into a computerized accounting log, while the personnel in the accounting department would use another program to print that log. As early user interfaces were relatively unrefined, it would not be unusual for those programs to refer to the accounting log and its data in very different ways.

As processor resources increased and interface techniques improved, programmers began writing software packages that operated simultaneously with one another and that displayed and printed shared data on more uniform bases. Developers of these early integrated packages attempted to insure that data files written by one program in a package could be read by another program in that package. Thus, for example, most packages permitted a table generated by a spreadsheet application to be read into the corresponding word processing application. In addition, many packages capitalized on object embedding technologies to "dynamically" link the data files generated by one application program into those of another. For example, a table generated by a spreadsheet application could be embedded in a word processing document so that changes made by the former were automatically included in the later.

A problem with prior art integrated packages is that they are not sufficiently integrated. For example, although data generated by one application can be dynamically linked into another, both applications typically cannot modify that data. Instead, current technologies restrict modification to the "source" application, i.e., the application that generated the data in the first instance. In addition, only the source application can format an embedded item. Thus, for example, the arrangement of a table that is dynamically embedded into a word processing document is restricted to the spreadsheet application that created that table.

An object of this invention is to provide improved digital data processing methodologies and, more particularly, improved methods for synchronizing processes executing on a digital data processing system.

A related object of the invention is to provide such methods as are suited for processes that share common data or that maintain corresponding data sets.

Another object upset of the invention use to permit such synchronization regardless of whether the processes are executing on the same computer or across networked computers.

Still another object of the invention is to provide methodologies that can be implemented at minimum cost for use with a broad range of applications program and across a wide spectrum of hardware and software platforms.

Summary of the Invention

The foregoing objects are among those attained on the invention which provides, in one aspect, a method for synchronizing multiple processes executing on a digital data processing system, e.g., a single computer or a plurality of computers that are in communication with one another.

The method includes registering each of the processes for notification of at least selected events occurring in the other processes. Those event types can include, for example, item-specific events (e.g., the addition, deletion or selection of an item in another process), menu selection events (i.e., the selection of the menu entry in another process), and the invocation or termination of another process. As used herein, "item" refers to any entity, such as a datum, used or displayed by a process. It can be, for example, an entry, row, or column in a spreadsheet program, an object displayed in a drawing program, a file opened by an editing program, etc.

The method further includes detecting an event in any of the multiple processes and determining whether that event is one for which processes (other than that in which the event occurred) are registered for notification. If so, those other processes are notified, e.g., so that they can take an action consistent with that caused by the detected event in the process in which it occurred.

By way of example, three applications programs executing simultaneously on a computer can register for notification of menu selections (or "picks") occurring in any of the others. When a user selects a menu options (such as FILE-OPEN) in any of the programs, the method notifies the others of that event. In response, they can interrogate a common database or the operating system to determine the specific action taken (e.g., the specific file opened) and, in turn, can take complimentary actions (e.g., opening related files).

By way of further example, an applications program can register for notification of process invocation or termination, e.g., when other programs are opened or exited. As new processes are started or old ones terminated, the registered process can, for example, alter the menu options it provides. More particularly, if a registered process has a menu with options that permit the user to open or switch to another process, that option can be activated or deactivated depending on whether that other process has already been invoked and is active, or whether it has been terminated and is inactive.

Further aspects of the invention provides methods as described above in which a registry, database, or other store is maintained identifying each process and the events of which it is to be notified.

Still further aspects of the invention provides methods as described above in which events occurring in processes are detected by intercepting selected function or subroutine calls made by those processes. This is accomplished with "shell" or "stub" routines having names like those of the functions/subroutines to be intercepted. Thus, for example, where it is known (e.g., as it is in most windowing environments) that processes typically invoke a specified routine in order to obtain menu selections from the user, an interceptor routine of the same name can be linked (e.g., via dynamic linking libraries, or DLL's) prior to linking of the specified routine. Once that interceptor routine is invoked, it can determine the type event and notify the other registered processes accordingly. Thereafter, the interceptor routine can itself invoke the specified routine, thereby, enabling the process in which the event occurred to continue processing in the normal course.

Other aspects of the invention provides methods as described above including the step of detecting data transfer events, such as "drag-and-drop" events, between any of the processes. Drag-and-drop events are events in which a user "drags" an item from the window of a first process to the window of a second process, thereby, causing that information to be added to the second process. Upon detection of a drag-and-drop event, the method determines whether the recipient of that operation is registered to receive data in that manner and, if so, at what site in its "window" and in what format. If the operation proceeds, the method that notifies other processes of the addition of an item to the recipient process. Those other processes can respond, in turn, by adding corresponding items to their own data sets.

Still other aspects of the invention contemplate systems operating in accord with the above-described methodologies for synchronizing multiple processes executing on a digital data processing apparatus.

Yet still other aspects of the invention contemplate articles of manufacture, e.g., magnetic disks, composed of computer usable media embodying a computer program that causes a digital data processing apparatus to synchronize multiple processes executing on such apparatus.

Methods and systems as described above can be advantageously used to synchronize multiple processes executing on a digital data processing system and, if desired, to insure the coherency of data maintained by them. The methods and systems permit all processes to be notified whenever a change is made to one of them so that, for example, as menu selections and item selections are made in one process, they can be mirrored in the other processes. And further, so that data sets independently maintained by each process can be updated for accord with those maintained by the other processes. And still further, so that as items are added to a process, or deleted therefrom, the other processes can update themselves accordingly.

Brief Description of the Drawings

A further understanding of the invention may be attained by reference to the drawings in which:

Figure 1 depicts a digital data processing system of the type in which the invention is practiced;

Figure 2 depicts initialization and termination sequences in a method according to the invention;

Figure 3 depicts a menu selection synchronization sequence in a method according to the invention;

Figure 4 depicts an item selection synchronization sequence in a method according to the invention;

Figure 5 depicts a drag-and-drop synchronization sequence in a method according to the invention;

Figures 6A - 6B illustrate the effect of menu selection synchronization in a method according to the invention;

Figures 7A - 7B illustrate the effect of item selection synchronization in a method according to the invention; and

Figures 8A - 8B illustrate the effect of drag-and-drop synchronization in a method according to the invention;

Figure 9 depicts an article of manufacture embodying a program intended to cause a computer to perform methods according to the invention.

Detailed Description of the Illustrated Embodiment

Operating Environment

Figure 1 depicts a digital data processing system 10 of the type in which the invention is practiced. The illustrated system 10 includes a conventional digital data processor 20 (e.g., a workstation, personal computer, hand-held computing device, etc.) having one or more central processing units 22, main memory 24, input-output system 26, and disk drive (or other mass storage device) 28, all of the conventional type known in the art

Digital data processor 20 is coupled to a monitor 29 for the display of output generated by applications programs, operating systems and other software executing thereon. Other output devices, such as printers and plotters (not shown), may also be

coupled to digital data processor 20. Likewise, input devices, such as a keyboard and "mouse" (not shown), may be attached to the unit 20 to accept user input.

Illustrated digital data processing system 10 further includes a conventional digital data processor 30, which is coupled to digital data processor 20 via network 34. The digital data processor 30 can also include central processing units, main memory, include-output systems, mass storage devices, monitors, output devices and input devices, as illustrated. The network 34 comprises any conventional digital data processing network (e.g., LAN or WAN), cable television-based network, wireless network and/or any telecommunications-based network capable of supporting communications between digital data processors 20, 30.

The drawing shows another conventional digital data processor 32 which is coupled to digital data processors 20, 30, but which does not execute processes that are synchronized by the invention. Rather, digital data processor 32 supplies data 40 that is displayed and/or manipulated by synchronized processes 42, 46, 48 on digital data processors 20, 30. That data 40 can represent, for example, information regarding a process, device, or combination thereof amenable to modeling or control. Data 40 can be, for example, operational information regarding a process control system of the type conventionally used to monitor and control a manufacturing process. Data 40 can also represent, by way of further example, configuration parameters for digital data processing network of the type used in a conventional corporate environment. Data 40 is supplied to processes 42, 46, 48 over network 34 in the conventional manner. A single copy of that data may reside in the stores of digital data processor 32, although, separate copies of that data can be maintained by each of processes 42, 46, 48 in the memories of their associated digital data processors 20, 30. As Figure 1 shows, processes 42, 46, 48 synchronized by the invention can operate on, or share, common data supplied by an external source. However, this is not a requirement of the invention -- which can synchronize processes regardless of their sources of data and regardless of whether that data is shared.

Synchronized Applications

Processors 20, 30, are programmed in accord with the teachings herein for synchronization of processes 42, 46, 48, executing thereon. In the illustrated embodiment, that synchronization is effected with respect to common data supplied, by way of example, from digital data processor 32. As noted above, the synchronization methodologies discussed herein can be applied regardless of whether the processes 42, 46, 48 operate on common data, corresponding data (i.e., related but not identical data sets), or the like.

As suggested by Figure 1, processes 42, 46, 48 execute in a "windowing" environment provided under the operating systems of the respective processors 20, 30. Although this is not essential to the invention, such environments can be implemented, for example, under the Windows 95 or Windows NT operating systems, as well as under a variety of other commercially available operating systems.

Within that environment, applications program 42 (executing on digital data processor 20) and applications program 46 (executing on digital data processor 30) display common data (i.e., from processor 32) in a first format. Applications program 48 (executing on digital data processor 30) displays that data in a second format. While those skilled in the art will appreciate that the particular display formats discussed herein and shown on monitors 29, 44 of Figure 1 are not essential to the invention, they help illustrate the respective functions of the processes 42, 46, 48 being synchronized.

Initialization/Termination of Synchronization Processes and Data Structures

Figure 2 is a modified data flow diagram illustrating initialization and termination of the data structures utilized for synchronization of processes 42, 46, 48 executing digital data processors 20, 30, respectively. Though only two of those processes 42, 46 are illustrated, those skilled in the art will appreciate that the methodologies shown in this and the other drawings, and discussed below, are

applicable to any number of other processes, whether they are executing on a single digital data processor or multiple digital data processors.

Referring to the drawing, process 42 comprises applications program 42a, as well as a menu manager 42b and an item manager 42c. The menu manager 42b works in conjunction with conventional functionality provided in the windowing API of the digital data processor 20 to display menus, obtain user menu selections and notify the other processes 46 of these selections.

For example, in an embodiment of the invention that utilizes the Windows 95 operating system, the menu manager 42b operates in connection with that system's menu resource data structure and the standard Windows 95 menu-generating routines. The menu manager 42b does not necessarily define a menu structure itself but, rather, utilizes the structures defined by the applications program 42b (and stored, for example, in the menu resource data structure).

At runtime, the menu manager 42b utilizes API functions supplied by the operating system to identify menus defined by the applications program 42a and to identify those selection events, i.e., menu "picks" made by the user. The menu manager then notifies the other processes of those events (as discussed below in connection with Figure 3).

In a preferred embodiment, the menu manager 42b does this by intercepting selected calls made to the windowing API by the applications program 42a. For example, in an embodiment of the invention that operates with Windows 95, the menu manager 42b incorporates a function named OnCmdMsg, that intercepts calls made by the applications program 42a to the similarly-named API function. That "interceptor" function is dynamically linked to the applications program 42a (e.g., via a DLL file) prior to the similarly-named API function.

Hence, at runtime the interceptor function is invoked prior to the API function that would otherwise be called by the applications program 42a. As discussed below,

once the interceptor function has been invoked, it notifies the other processes 46 of the menu selection events in process 42 and, particularly, in applications program 42a.

The menu manager 42b also adds, to applications program 42a, menu options for every applications program (i.e., other than program 42a) that will serve as a menu selection for program 42a. Those other applications programs are registered for this purpose by providing, for each of them, a companion file that has the same basic filename as the executable form of the registered program but with a preselected filename extension (e.g., ".mnu"). The menu manager detects those companion files (e.g., by searching for their ".mnu" extensions) and adds corresponding menu options to applications program 42a. In each case, the added menu option consists of the contents of the companion file. If the user selects any of those options at runtime, the menu manager invokes the corresponding executable file.

Thus, for example, if the programs EDITOR and CATALOG will serve as menu selections for program 42a, they are registered by supplying companion files "editor.mnu" and "catalog.mnu". Upon identifying those files, the menu manager 42b adds menu options "editor" and "catalog" to the menu of applications program 42a. If either of those options is selected, a corresponding executable file name is constructed from the option name and the appropriate executable extension, e.g.,"editor.exe" and "catalog.exe" in the case of Windows 95, and that file is executed.

With reference to Figure 2, in step (1), the applications program 42a initializes the menu manager 42b by invoking its InitializeManager entry point. This causes the menu manager 42b to initialize data structures used at runtime, particularly, the process/data synchronization structure 44, which is a common data structure utilized by all processes 42, 46, 48 synchronized by the invention. That structure 44 lists all of the synchronized processes and the types of events of which they are to be notified. Those events can include, for example, menu selection events (i.e., the selection of menu options in a process) and the invocation or termination of another process.

In a preferred embodiment of the invention, the process/data synchronization structure 44 has a structure that is defined as follows:

- process identification
- process type
- for each notification type, the type of notification desired

Upon being initialized by the applications program 42a, the menu manager 42b updates the process/data synchronization structure to register that process for notification of menu selection events occurring in the other synchronized processes, e.g., process 46. At the same time, the menu manager 42b employs the item manager 42c to include an instance of program 42a in the process/data synchronization structure. Other processes can adjust their own menus using the menu manager and that structure's information, e.g., deactivating menu selections that could otherwise be used to initiate program 42a.

With continued reference to Figure 2, item manager 42c works in conjunction with conventional functionality provided in the windowing API of the digital data processor 20 to identify item specific events in the applications program 42a and to notify the other processes 46 of the same.

In an embodiment of the invention for use with the Windows 95 operating system, the item manager 42c operates in connection with that system's standard API notification input routines, i.e., the routines that identify items acted upon by the user in the window of the applications program 42a. Unlike the menu manager, which intercepts calls made by the applications program 42a to the windowing system, the item manager 42c is directly invoked by the applications program as each item-specific event is detected. Thus, for example, the applications program 42a invokes the item manager 42c upon each item selection, deselection, addition, deletion or modification.

With continued reference to Figure 2, in step (2), the applications program 42a initializes the item manager 42b by invoking its InitializeManager entry point. More particularly, the applications program 42a invokes that entry point and specifies whether

the program is to be notified of item-specific events, to wit, the addition, deletion or selection of an item in another process. The applications program 42a also specifies whether that program 42a may receive data via data transfer operations, such as drag-and-drop and/or cut-and-paste operations, and, if so, the permissible format and "drop" site (in the case of drag-and-drop) for such operations. As those skilled in the art will appreciate, a "drag-and-drop" is a common windowing operation wherein data graphically selected in one applications program is"dragged" to another applications program and, thereby, added to that other program. The"drop" site is the location in the recipient applications program to which an icon representing the data is dragged. A "cut-and-paste" is a common windowing operation wherein data selected in one applications program is copied (e.g., via a keyboard or mouse command) to a "clipboard" and, subsequently, "pasted" into another applications program.

Item manager 42c stores data transfer information for each process in a common data structure referred to as the transfer site structure 45. In a preferred embodiment of the invention, that structure is defined as follows:

- process identification
- region identification
- ownership indicator
- type of data transfers allowed (e.g., drag-and-drop, cut-and-paste, etc.)

In steps (3)-(4), the applications program 46a of process 46 similarly invokes its respective menu manager 46b and item manager 46c to register the program 46a for notification of desired events, e.g., menu selection events, item-specific events, and drag-and-drop events. As above, that registration information is stored in the common data structures 44 and 45.

The processes 42, 46 can be de-registered from synchronization by transmission of termination notice to their respective item managers 42c, 46c. This causes entries for the respective processes 42, 46 to be removed from the shared structures data structures 44, 45, thus, obviating their notification of further events occurring in the other processes. This is indicated by steps (5) - (6) in the drawing. Upon termination

of an applications program, the item managers can notify all other processes of termination of the process. The notified processes could adjust their menu selections accordingly, e.g., by activating or deactivating menu options to invoke the terminated process.

Synchronization via Menu Selection

Figure 3 depicts a sequence for synchronizing processes 42, 46 based on a menu selection event occurring in one of the processes, to wit, application program 42a. In step (1), the menu manager 42b detects a menu selection event arising from the user's selection of a menu item in the applications program 42a. Such a selection is intercepted by the menu manager 42b in the manner discussed above.

An identification of the menu option chosen by the user is reflected by a value MENU_ID intercepted by the menu manager 42b. In a preferred embodiment of the invention, MENU_ID has a commonly-known value representing the type of menu selection chosen by the user, e.g., FILE-NEW, FILE-OPEN, FILE-CLOSE. Those skilled in the art will, of course, appreciate that MENU_ID may take on other values reflecting that a menu selection event occurred and/or the type of that event.

In step (2), the menu manager 42b passes the MENU_ID to the item manager 42c for possible notification of the other applications programs 46, 48. A preferred menu manager 42b of the invention does not pass MENU_ID's for all intercepted menu selection events to item manager 42c. Rather, it only passes those identifications for menu selections that effect the opening or closing of files, such as FILE-NEW, FILE-OPEN, FILE-CLOSE. Those skilled in the art will, of course, appreciate that MENU_ID's for all or other menu selection events occurring in applications program 42a can be passed to item manager 42c.

In step (3), the item manager 42c determines which processes -- other than that in which the menu selection event was detected -- are registered for notification of menu selection events. This is determined from the process/data synchronization

structure 44, which identifies processes registered for notification of such events, as discussed above.

In step (4), the item manager 42c notifies each of the interested registered processes 46, 48 of the detected menu selection event. Particularly, the item manager transmits the MENU_ID value to each of the registered applications programs 46a, 48a. Such transmission can be accomplished using traditional inter-process communication (IPC) techniques, such as provided by the Windows 95 function SendMessage. Although all of the registered processes can be identified simultaneously, e.g., by a "broadcast" communication, a preferred embodiment of the invention notifies the registered processes serially. Each notified applications program 46a, 48a can respond to the notification, e.g., by opening or closing a corresponding file, creating a corresponding database, etc.

Once the registered processes have been notified, the menu manager 42b permits the process in which the menu selection event was detected (i.e., applications program 42a) to continue. To this end, menu manager 42b invokes the windowing API function that was originally called by that program 42a but that was intercepted by the same-named menu manager interceptor routine.

Synchronization via Item Selection, Addition, Deletion

Figure 4 depicts a sequence for synchronizing processes 42, 46 based on an item-specific event (e.g., an item selection, addition or deletion) occurring in one of the processes, to wit, application program 46a. As noted above, the methods described herein can be extended to the other processes, e.g., process 48, as well.

In step (1), the applications program 42a invokes item manager 42c to register the program 42a for notification of specified item-specific events. The illustrated embodiment permits such an applications program to register for item selection, addition or deletion. Other embodiments of the invention can permit registration for notification of other item-specific events, such as highlighting, inactive/active state, etc.

In step (2), the item manager 42c stores in the process/data synchronizing structure 44 an identification of the process 42 (or of the applications program 42a), along with an indication of the types of item-specific events for which it is to be notified. The process/data synchronization structure 44 is defined as described above.

In step (3), the item manager 46c associated with process 46 detects an item-specific event (e.g., an item selection, addition or deletion) in corresponding applications program 46a. As discussed above, such detection is based on interception of routines in the windowing API of the operating system that identify objects selected, added or deleted by the user in the applications program 46a.

An identification of the specific item chosen by the user is reflected by a value ITEM_ID assigned and transmitted by the applications program 46a to the item manager 46c. In a preferred embodiment of the invention, ITEM_ID has a unique value representing the specific item chosen by the user or assigned directly by the applications program 46a. Those skilled in the art will, of course, appreciate that ITEM_ID may be assigned in other ways, preferably, that uniquely identify each item among all of the registered processes.

In step (4), the item manager 46c determines which processes -- other than that in which the item-specific event was detected -- are registered for notification of item-specific events. This is determined from the process/data synchronization structure 44, which (as discussed above) identifies processes registered for notification of such events.

In step (5), the item manager 46c notifies each of the registered processes of the detected item specific event. Particularly, the item manager 46c transmits the ITEM_ID(s) affected by the event to each of the registered applications programs 42a. Such transmission can be accomplished using traditional inter-process communication techniques, as discussed above. Each notified applications program 42a can respond to the notification, e.g., by selecting, adding or deleting the corresponding item in its own display window.

In embodiments where the applications programs maintain respective data stores reflecting, e.g., the content of a common data store, e.g., data store 40 of digital data processor 32 (Figure 1), a notified applications program 42a can respond to notification to update its respective data store.

Once the registered processes have been notified, the item manager 46b permits the process in which the item-specific event was detected (i.e., applications program 46a) to continue.

Synchronization via Data Transfer Operations

Figure 5 depicts a sequence for synchronizing processes 42, 46 based on data transfer events, such as a drag-and-drop events and/or cut-and-paste events, occurring between two of the processes, to wit, applications program 48a and applications program 46a.

At the outset, each applications program that is a potential recipient of a data transfer operation registers with its respective item manager (i) the types of data transfers that program can receive (e.g., drag-and-drop, cut-and-paste, etc.), (ii) the preferred format in which data is to be transferred, and (iii) for programs registered for drag-and-drop, the valid drop sites. This is illustrated in step(0), wherein applications program 46a registers with item manager 46c to receive drag-and-drop transfers.

In steps (1) - (2), the other applications programs, e.g., 42a, utilizes their corresponding item managers, e.g., 42c, to register for notification of item-specific events. This ensures that the program 42a will be notified of item additions resulting from drag-and-drop data transfers received by other programs, e.g., 46a.

In step (3), the item manager 48c associated with the applications program 48a in which a drag-and-drop event is initiated detects the start of that event. Such detection is based on interception of mouse "down" and mouse "move" notifications by that program 48a from the windowing operating system.

In step (4), the item manager 48c confirms the drop site selected by the user for the drag-and-drop event. As discussed above, valid drop sites for each registered process are stored in the transfer site structure 45. Comparing the information in that instruction with the information supplied by the operating system vis-a-vis the identity of the recipient process 46 and the drop site selected by the user, the item manager 48c permits or aborts the drag-and-drop event. In the event that the event is permitted to continue, the item manager 48c formats the transferred data in the format specified for the recipient program 46a in the transfer site structure 45. The formatted data is then transferred using traditional inter-process communication (IPC) techniques (e.g., SendMessage) of the type discussed above. Once that data is received by the recipient, to wit, applications program 46c, it adds the specified items in the format provided.

Once the data transfer is completed, the item manager 46c associated with the recipient applications program 46a notifies the other processes of the additional item. This is accomplished in steps (5) - (7) of Figure 5 in the same manner discussed above in connection with steps (3) - (5) of Figure 4.

Though the illustrated embodiment concerns drag-and-drop events, those skilled in the art will appreciate that it can be readily applied to other types of inter-applications data transfers, for example, cut-and-paste transfers (e.g., where selected item(s) in a first application are "clipped" to the clipboard and, subsequently, copied into a second application).

Synchronization Illustrated

Figures 6 - 9 depict screen dumps illustrating effects of synchronizing processes in accord with the board's discussed above.

Referring to Figures 6a - 6b, there is shown the effects of synchronizing applications program 42a based on a menu selection event occurring in another applications program 46a. More particularly, as shown in Figure 6a, the two applications programs 42a, 46a are executing simultaneously on a digital data

processing system, e.g., in the manner of the processes executing on processor 30 of Figure 1. With continued reference to Figure 6a, the user (not shown) selects a menu option -- particularly, a FILE-OPEN option -- in applications program 46a. In accord with the methodologies discussed above, e.g., in connection with Figure 3, applications program 42a is notified of the selection. As shown in Figure 6b, that applications program 42a responds to the notification by opening a corresponding file and displaying its data.

Referring to Figures 7a - 7b, there is shown the effects of synchronizing applications programs 46a, 50a based on an item selection event occurring in applications program 42a. More particularly, as shown in Figure 7a, the three applications programs 42a, 46a, 50a are executing simultaneously on a digital data processing system. Those programs are also executing with respect to common data, although they display it in differing formats. As shown in Figure 7a, the user (not shown) selects two items 52, 54 displayed by applications program 42a. In accord with the methodologies discussed above, e.g., in connection with Figure 4, applications programs 46a, 50a are notified of the selection. As shown in Figure 7b, those applications programs 46a, 50a respond to the notification by selecting or highlighting corresponding items displayed by them.

Referring to Figures 8a - 8b, there is shown the effects of synchronizing applications program 46a based on a drag-and-drop event occurring between two other processes 52a, 42a. More particularly, as shown in Figure 8a, the three applications programs 42a, 46a, 52a are executing simultaneously on a digital data processing system. As shown in Figure 8a, the user (not shown) drags items 54 displayed by applications program 52a to applications program 42a. In accord with the methodologies discussed above, e.g., in connection with Figure 5, the items are formatted and the drop site confirmed before it is transferred to the recipient applications program 42a. Moreover, the applications program 46a is notified of the event, particularly, of the addition of items 54' single prime, not double prime to applications program 42a. As shown in Figure 8b, that applications program 46 responds to the notification by adding a corresponding items 54' to its display.

Articles of Manufacture

Figure 9 depicts an article of manufacture, to wit, a magnetic diskette, composed of a computer usable media, to wit, a magnetic disk, embodying a computer program that causes digital data processing system 10, or other such digital data processing apparatus, to operate in accord with the methods described above. The diskette is shown in front view and back view. It is of conventional construction and has the computer program stored on the magnetic media therein in a conventional manner readable, e.g., via a read/write head contained in a diskette drive of image analyzer 40. It will be appreciated that diskette is shown by way of example only and that other articles of manufacture comprising computer usable media on which programs intended to cause a computer to execute in accord with the teachings hereof are also embraced by the invention.

Conclusion

Described herein are improved methods, systems and articles of manufacture for digital data processing meeting the objects set forth above. It will be appreciated that the embodiments illustrated and discussed herein are merely examples of the invention and that other embodiments incorporating modifications thereto fall within the scope of the invention, of which we claim:

1. An event-driven method for synchronizing plural processes on a digital data processing system, the method comprising

A. registering each of the plural processes for notification of at least selected events in the other processes;

B. detecting an event in any of the plural processes,

C. determining whether the detected event is one for which a process, other than that in which the event occurred, is registered for notification; and

D. responding to an affirmative determination in step (C) for notifying the other process of the detected event.

2. A method according to claim 1, comprising the step of selectively updating that other process to reflect a change corresponding to that effected by the detected event in the process in which the event occurred.

3. A method according to claim 2, wherein the updating step includes any of the steps of (i) adding within that other process a datum added to the process in which the event occurred; (ii) deleting from that other process a datum deleted from the process in which the event occurred; (iii) selecting within that other process an item selected in the process in which the event occurred; and (iv) modifying a menu within that other process.

4. A method according to claim 3, wherein step (iv) comprises the step of responding to any of process invocation and process termination for any of activating and deactivating a menu option within that other process.

5. A method according to claim 1, wherein step (A) comprises the step of maintaining a registry of processes and events of which they are to be notified.

6. A method according to claim 5, wherein step (A) includes registering at least selected processes for notification of an event including any of (i) addition of a datum to another process, (ii) deletion of a datum from another process, (iii) selection of a datum in another process, (iv) selection of a menu option in another process, (v) invocation of another process, and (vi) termination of another process.

7. A method according to claim 1, wherein the digital data processing system comprises a plurality of interconnected digital data processors and wherein the plural processes are executing on one or more of those digital data processors.

8. A method according to claim 1, wherein step (D) comprises the step of notifying that other process of (i) an identity of the process in which the detected event occurred and (ii) a type of the detected event.

9. A method according to claim 1, wherein step (B) comprises the step of detecting an event in a process by intercepting a call issued by that process with a routine having a name like that which the intercepted call is directed.

10. A method according to claim 9, wherein step (B) comprises the step of generating a call to the routine to which the intercepted call was directed, after notifying any other processes of the detected event.

11. An event-driven method for synchronizing plural processes on a digital data processing system, the method comprising

A. registering at least selected processes for notification of menu selection events in the other processes;

B. detecting a menu selection event in any of the plural processes,

C. identifying processes, other than that in which the menu selection event was detected, registered for notification of menu selection events and notifying them of the detected event.

12. A method according to claim 11, comprising the step

(D) responding to such notification for selectively executing, within each of the identified processes, an operation in accord with that executed in connection with the detected menu selection event in the process in which it was detected.

13. A method according to claim 12, comprising the step of responding to notification of file access-related menu selection event for executing, within each of the identified processes, an operation for any of initiating access, accessing, and terminating access to a file corresponding to that effected by the file access-related menu selection event in the process in which it was detected.

14. A method according to claim 13, comprising the step of responding to notification of a file-open menu selection event for accessing, within each of the identified processes, a file corresponding to that opened by the menu selection event in the process in which it was detected.

15. A method according to claim 13, comprising the step of responding to notification of a file-close menu selection event for terminating access, within each of the identified processes, to a file corresponding to that to which access is terminated by the menu selection event in the process in which it was detected.

16. A method according to claim 12, wherein step (B) comprises the step of detecting a menu selection event in a process by intercepting a call issued by that process with a routine having a name like that which the intercepted call is directed, and wherein the method further comprises the step of invoking, after step (D), the routine to which the intercepted call was directed.

17. A method according to claim 11, wherein step (C) comprises the step of transferring to each of the identified processes a menu selection signal identifying the detected menu selection event.

18. A method according to claim 11, wherein the digital data processing system comprises a plurality of interconnected digital data processors and wherein the plural processes are executing on one or more of those digital data processors.

19. A method according to claim 11, wherein step (C) comprises the step of notifying the processes registered for notification of menu selection events of (i) an identity of the process in which the menu selection event was detected, (ii) a type of the menu selection event.

20. A method according to claim 19, wherein step (C) comprises the step of notifying the processes registered for notification of menu selection events of menu selection events for any of initiating and terminating access to a file.

21. An event-driven method for synchronizing plural processes on a digital data processing system, the method comprising

A. registering at least selected processes for notification of item-specific events in the other processes;

B. detecting an item-specific event in any of the plural processes,

C. identifying processes, other than that in which the item-specific event was detected, registered for notification of item-specific events and notifying them of the detected event.

22. A method according to claim 21, in which step (B) includes the step of detecting as an item-specific event any of a selection of an item, creation of an item, updating a value of an item, and deletion of an item.

23. A method according to claim 22, comprising the step of responding to such notification for taking action on an item within each of the identified processes in accord with that taken on a corresponding item in connection with the detected item-specific event in the process in which it was detected.

24. A method according to claim 23, comprising the step of responding to such notification for any of selecting, adding, updating and deleting an item in any of the identified processes corresponding to an item effected by the detected item-specific event was taken in the process in which it was detected.

25. A method according to claim 21, wherein the digital data processing system comprises a plurality of interconnected digital data processors and wherein the plural processes are executing on one or more of those digital data processors.

26. A method according to claim 21, wherein step (C) comprises the step of notifying the identified processes of (i) an identity of the process in which the item-specific selection event was detected, (ii) a type of the item-specific selection event.

27. A method according to claim 21, comprising the steps of

detecting a data transfer event between any of the processes;

identifying processes, other than a recipient of the data transfer event, registered for notification of item-specific events and notifying them of addition of an item to a process; and

responding to such notification for taking action within each of the identified processes in accord with the addition of an item to the recipient of the data transfer event.

28. A method according to claim 27, wherein the data transfer event includes any of a drag-and-drop event and a copy-and-paste event.

29. A method according to claim 21, comprising the steps of

detecting a drag-and-drop event between any of the processes;

identifying processes, other than a recipient of the drag-and-drop event, registered for notification of item-specific events and notifying them of addition of an item to a process; and

responding to such notification for taking action within each of the identified processes in accord with the addition of an item to the recipient of the drag-and-drop event.

30. A method according to claim 29, comprising the steps of

registering at least selected processes as recipients of drag-and-drop events and identifying valid drop sites within those processes; and

responding to detection of a drag-and-drop event for verifying validity of a drop site for that event in accord with the registration of recipients thereof.

31. A system for event-driven synchronization of plural processes on digital data processing apparatus, the system comprising

A. means for registering each of the plural processes for notification of at least selected events in the other processes;

B. means for detecting an event in any of the plural processes,

C. means for determining whether the detected event is one for which a process, other than that in which the event occurred, is registered for notification; and

D. means for responding to an affirmative such determination for notifying the other process of the detected event.

32. A system according to claim 31, comprising means for selectively updating that other process to reflect a change corresponding to that effected by the detected event in the process in which the event occurred.

33. A system according to claim 31, wherein the means for registering maintains a registry of processes and events of which they are to be notified.

34. A system according to claim 31, wherein the digital data processing apparatus comprises a plurality of interconnected digital data processors and wherein the plural processes are executing on one or more of those digital data processors.

35. A system according to claim 31, wherein the means for responding notifies the other process of (i) an identity of the process in which the detected event occurred and (ii) a type of the detected event.

36. A system according to claim 1, wherein the means for detecting detects an event in a process by intercepting a call issued by that process with a routine having a name like that which the intercepted call is directed.

37. A system for event-driven synchronization of plural processes on a digital data processing apparatus, the system comprising

A. means for registering at least selected processes for notification of menu selection events in the other processes;

B. means for detecting a menu selection event in any of the plural processes,

C. means for identifying processes, other than that in which the menu selection event was detected, registered for notification of menu selection events and notifying them of the detected event.

38. A system according to claim 37, comprising

D. means for responding to such notification for selectively executing, within each of the identified processes, an operation in accord with that executed in connection with the detected menu selection event in the process in which it was detected.

39 A system according to claim 37, wherein the means for identifying transfers to each of the identified processes a menu selection signal identifying the detected menu selection event.

40. A system according to claim 37, wherein the digital data processing apparatus comprises a plurality of interconnected digital data processors and wherein the plural processes are executing on one or more of those digital data processors.

41. A system according to claim 37, wherein the means for identifying notifies the processes registered for notification of menu selection events of (i) an identity of the process in which the menu selection event was detected, (ii) a type of the menu selection event.

42. A system for event-driven synchronization of plural processes on a digital data processing apparatus, the system comprising:

A. means for registering at least selected processes for notification of item-specific events in the other processes;

B. means for detecting an item-specific event in any of the plural processes,

C. means for identifying processes, other than that in which the item-specific event was detected, registered for notification of item-specific events and notifying them of the detected event.

43. A system according to claim 42, in which the means for detecting detects as an item-specific event any of a selection of an item, creation of an item, updating a value of an item, and deletion of an item.

44. A system according to claim 42, comprising means for responding to such notification for taking action on an item within each of the identified processes in accord with that taken on a corresponding item in connection with the detected item-specific event in the process in which it was detected.

45. A system according to claim 42, wherein the digital data processing apparatus comprises a plurality of interconnected digital data processors and wherein the plural processes are executing on one or more of those digital data processors.

46. A system according to claim 42, wherein the means for identifying notifies the identified processes of (i) an identity of the process in which the item-specific selection event was detected, (ii) a type of the item-specific selection event.

47. A system according to claim 42, comprising

means for detecting a data transfer event between any of the processes;

means for identifying processes, other than a recipient of the data transfer event, registered for notification of item-specific events and notifying them of addition of an item to a process; and

means for responding to such notification for taking action within each of the identified processes in accord with the addition of an item to the recipient of the data transfer event.

48. A system according to claim 42, comprising

means for detecting a drag-and-drop event between any of the processes;

means for identifying processes, other than a recipient of the drag-and-drop event, registered for notification of item-specific events and notifying them of addition of an item to a process; and

means for responding to such notification for taking action within each of the identified processes in accord with the addition of an item to the recipient of the drag-and-drop event.

49. An article of manufacture comprising a computer usable medium embodying program code for causing a digital data processor to carry out a method of synchronizing plural processes on a digital data processing system, the method comprising

A. registering each of the plural processes for notification of at least selected events in the other processes;

B. detecting an event in any of the plural processes,

C. determining whether the detected event is one for which a process, other than that in which the event occurred, is registered for notification; and

D. responding to an affirmative determination in step (C) for notifying the other process of the detected event.

50. An article of manufacture comprising a computer usable medium embodying program code for synchronizing plural processes on a digital data processing system, the method comprising

A. registering at least selected processes for notification of menu selection events in the other processes;

B. detecting a menu selection event in any of the plural processes,

C. identifying processes, other than that in which the menu selection event was detected, registered for notification of menu selection events and notifying them of the detected event.

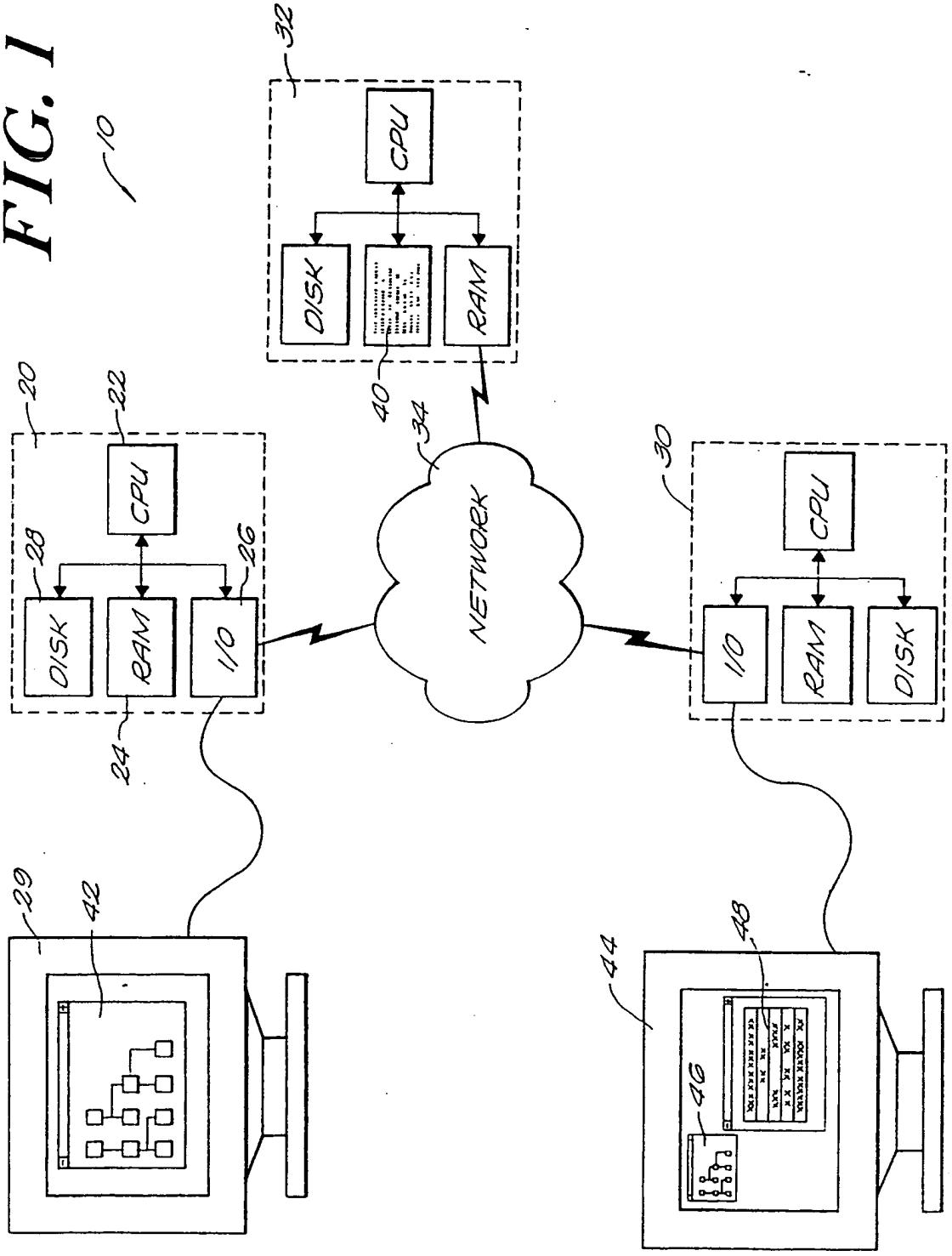
51. An article of manufacture comprising a computer usable medium embodying program code for synchronizing plural processes on a digital data processing system, the method comprising

A. registering at least selected processes for notification of item-specific events in the other processes;

B. detecting an item-specific event in any of the plural processes,

C. identifying processes, other than that in which the item-specific event was detected, registered for notification of item-specific events and notifying them of the detected event.

FIG. 1



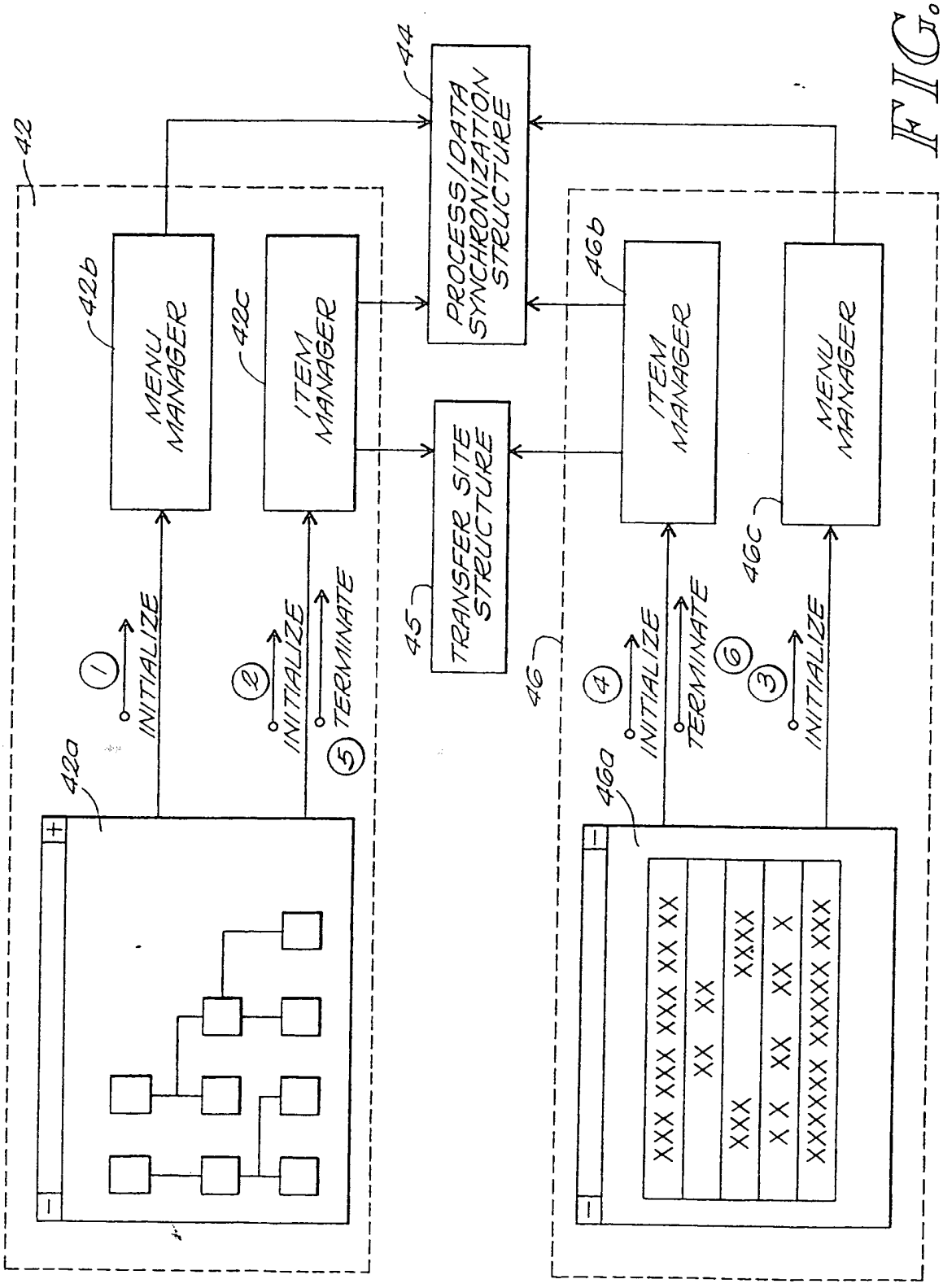


FIG. 2

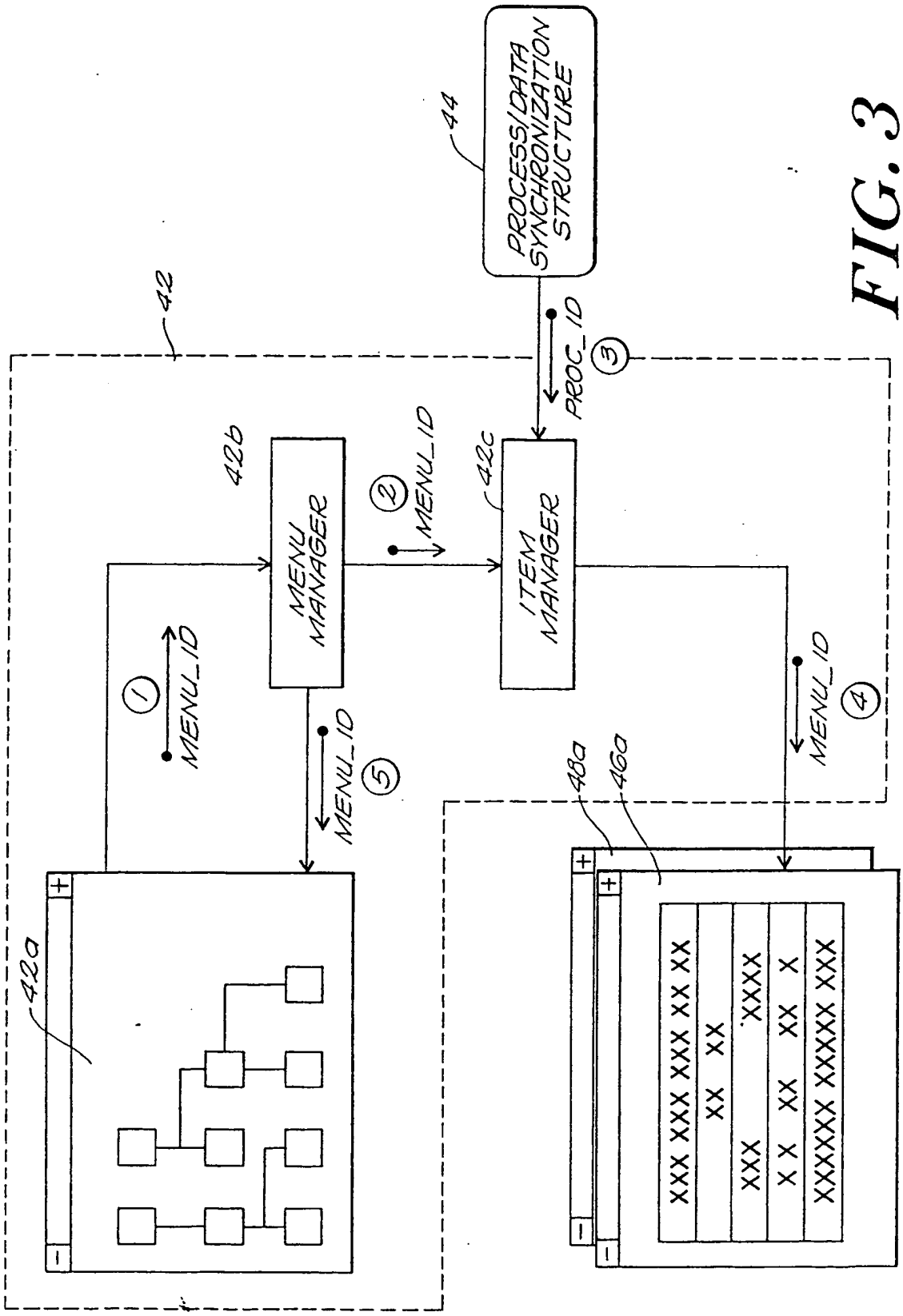


FIG. 3

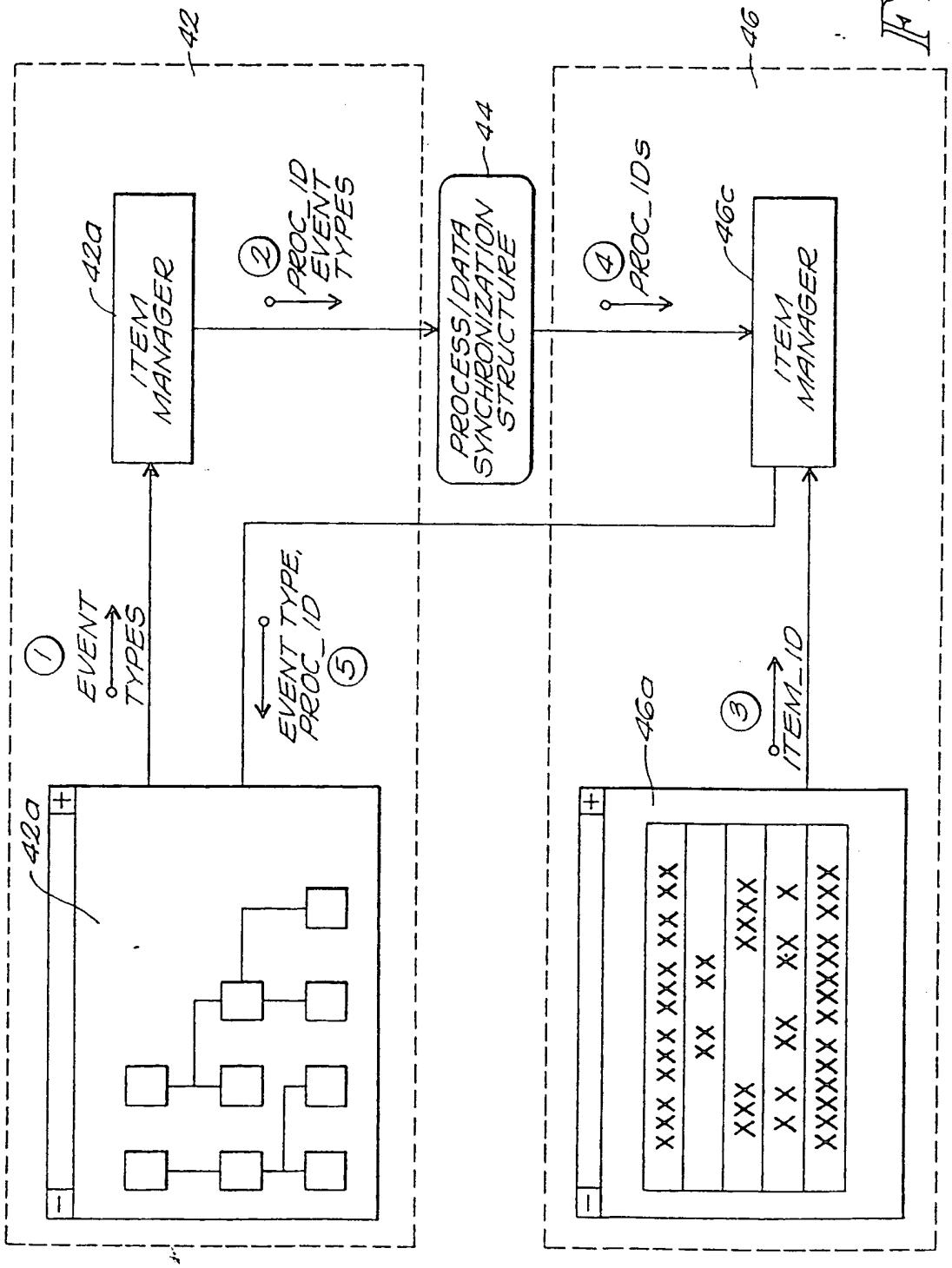


FIG. 4

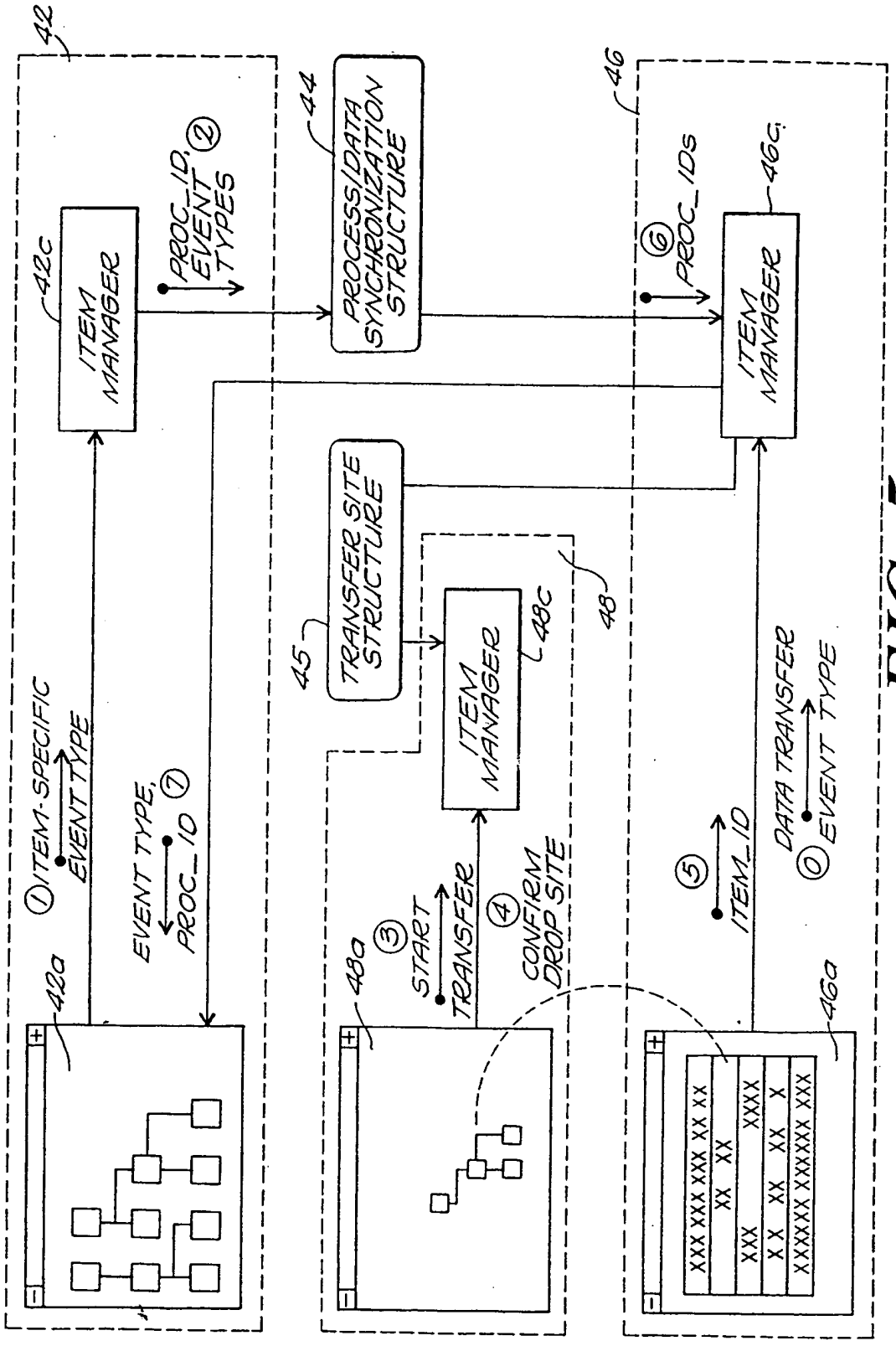


FIG. 5

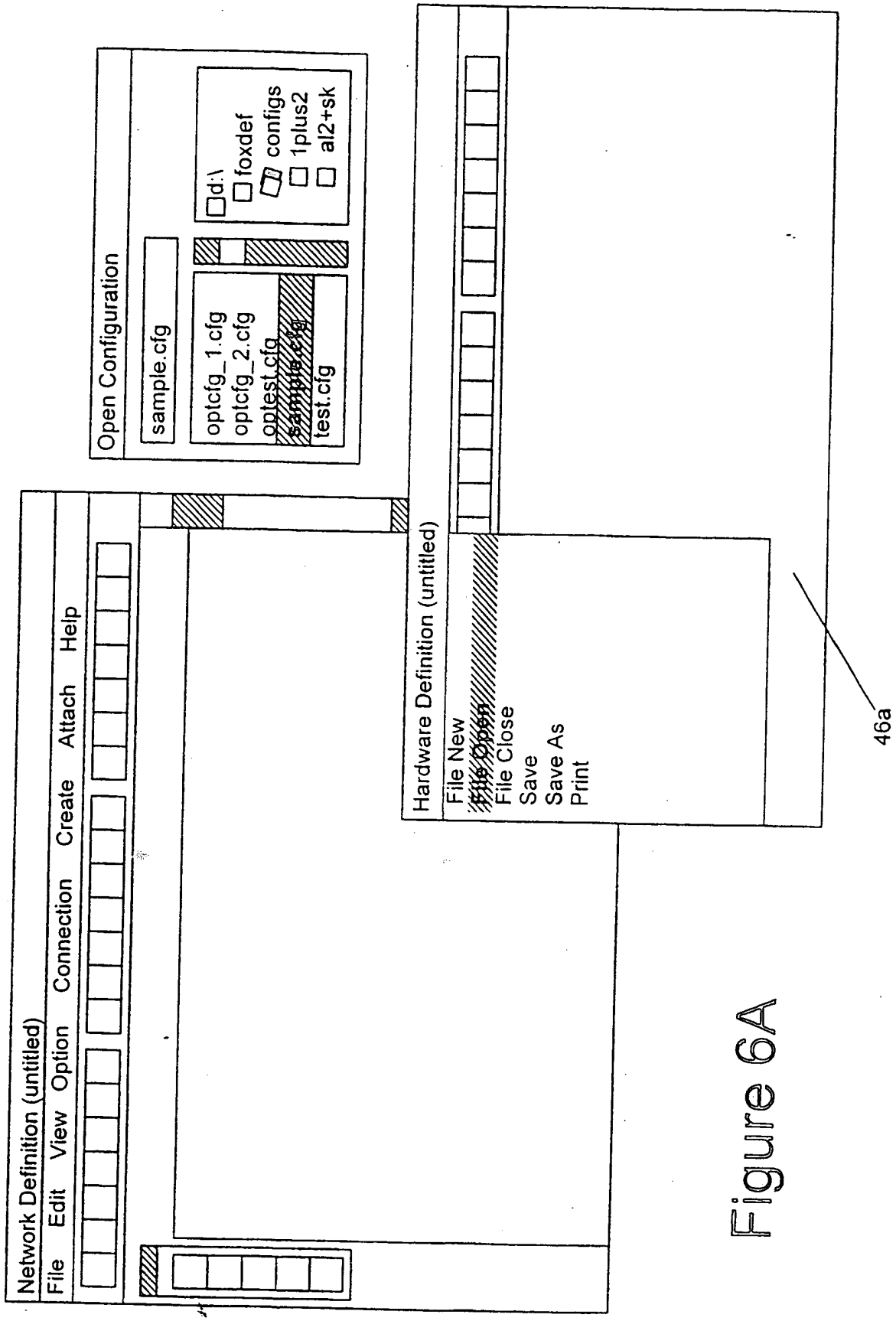


Figure 6A

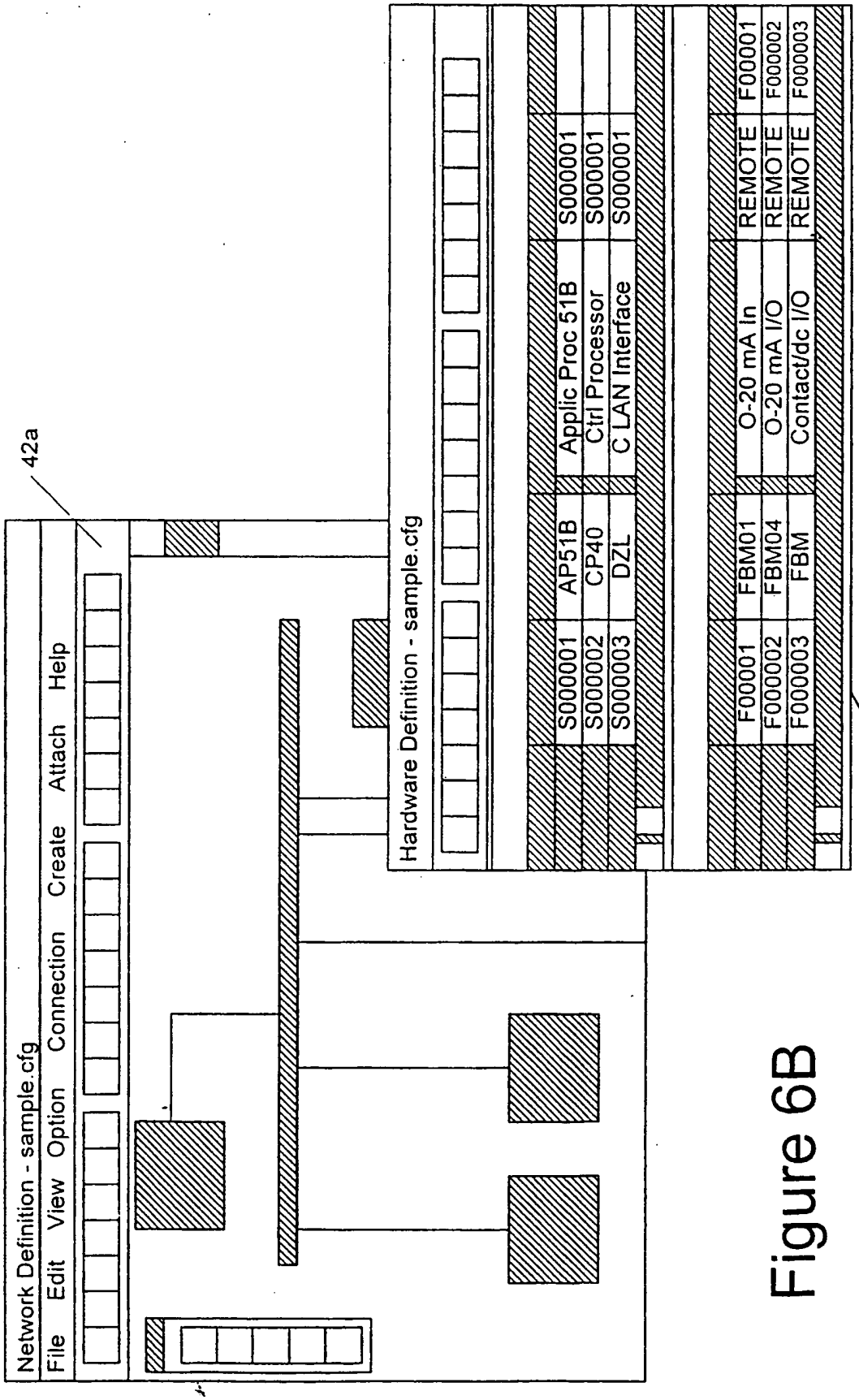


Figure 6B

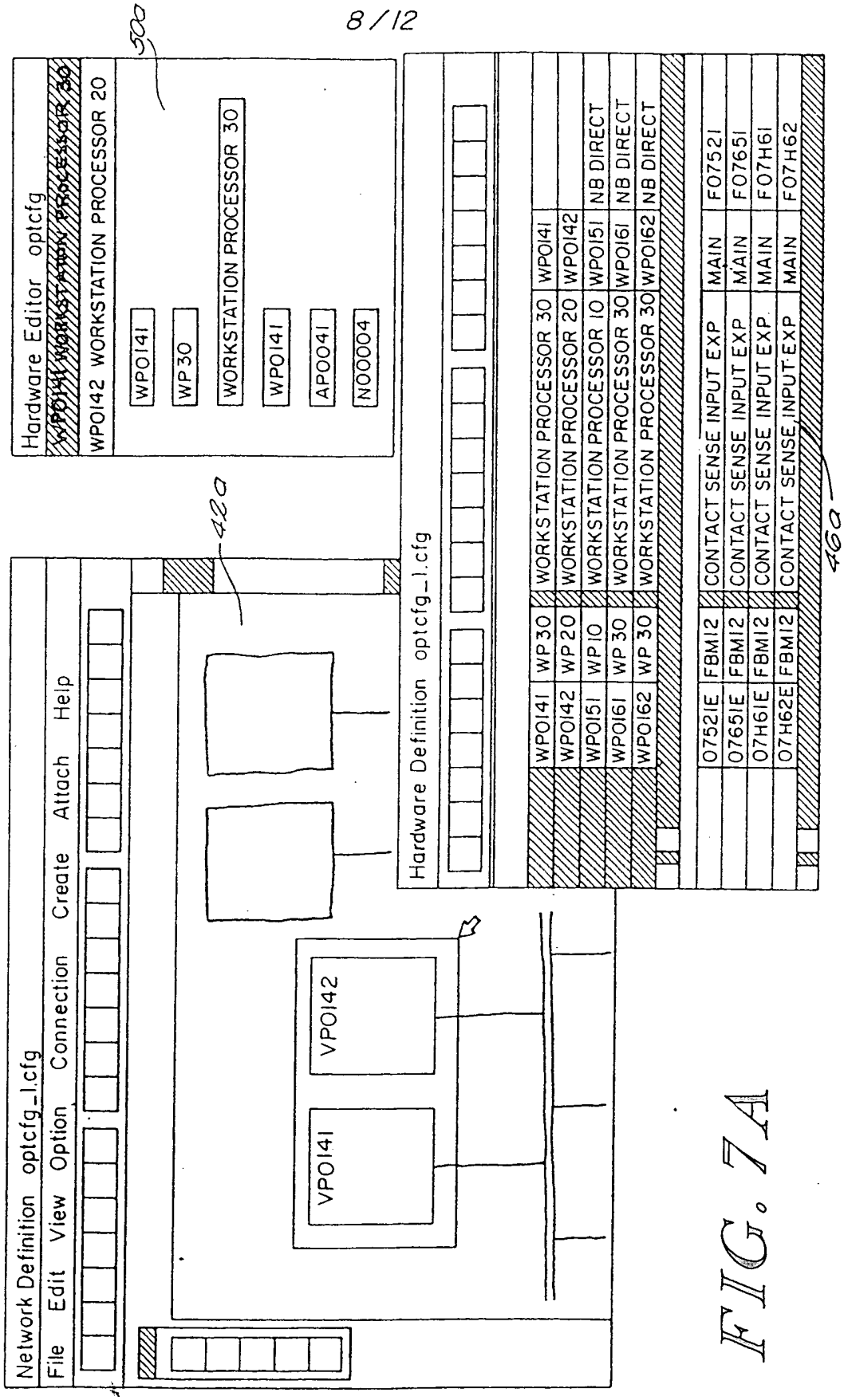


FIG. 7A

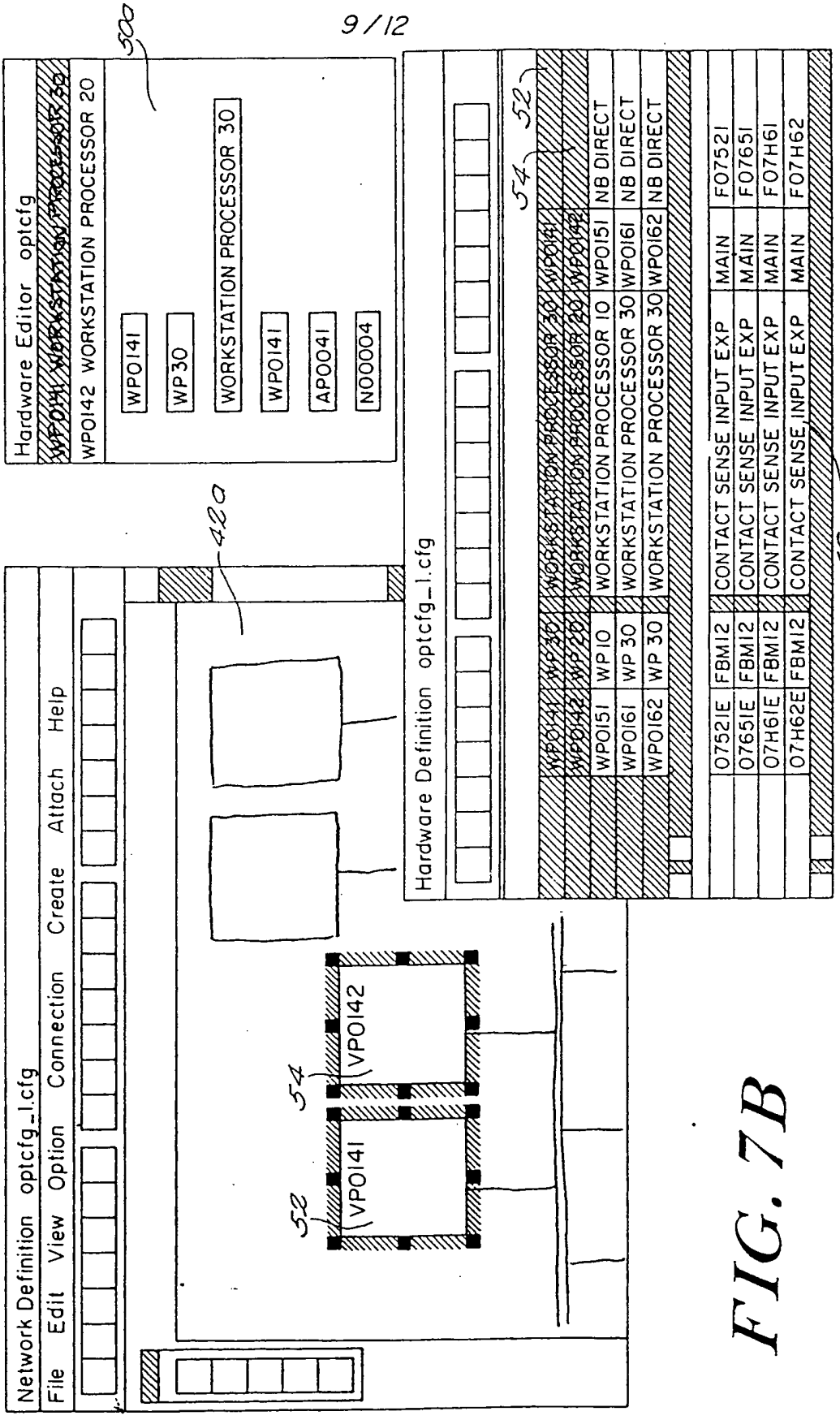

Network Definition optcfg_1.cfg
File Edit View Option Connection Create Attach Help
VP0141
VP0142
52
54
420
Hardware Definition optcfg_1.cfg
Hardware Editor optcfg
WP0142 WORKSTATION PROCESSOR 20
500
WP0141
WP30
WORKSTATION PROCESSOR 30
WP0141
AP0041
N00004
52
54
460
WP0141 WP30 WORKSTATION PROCESSOR 30 WP0141
WP0142 WP20 WORKSTATION PROCESSOR 20 WP0142
WP0151 WP10 WORKSTATION PROCESSOR 10 WP0151 NB DIRECT
WP0161 WP 30 WORKSTATION PROCESSOR 30 WP0161 NB DIRECT
WP0162 WP 30 WORKSTATION PROCESSOR 30 WP0162 NB DIRECT
0752IE FBMI2 CONTACT SENSE INPUT EXP MAIN F07521
0765IE FBMI2 CONTACT SENSE INPUT EXP MAIN F07651
07H61E FBMI2 CONTACT SENSE INPUT EXP MAIN F07H61
07H62E FBMI2 CONTACT SENSE INPUT EXP MAIN F07H62

FIG. 7B

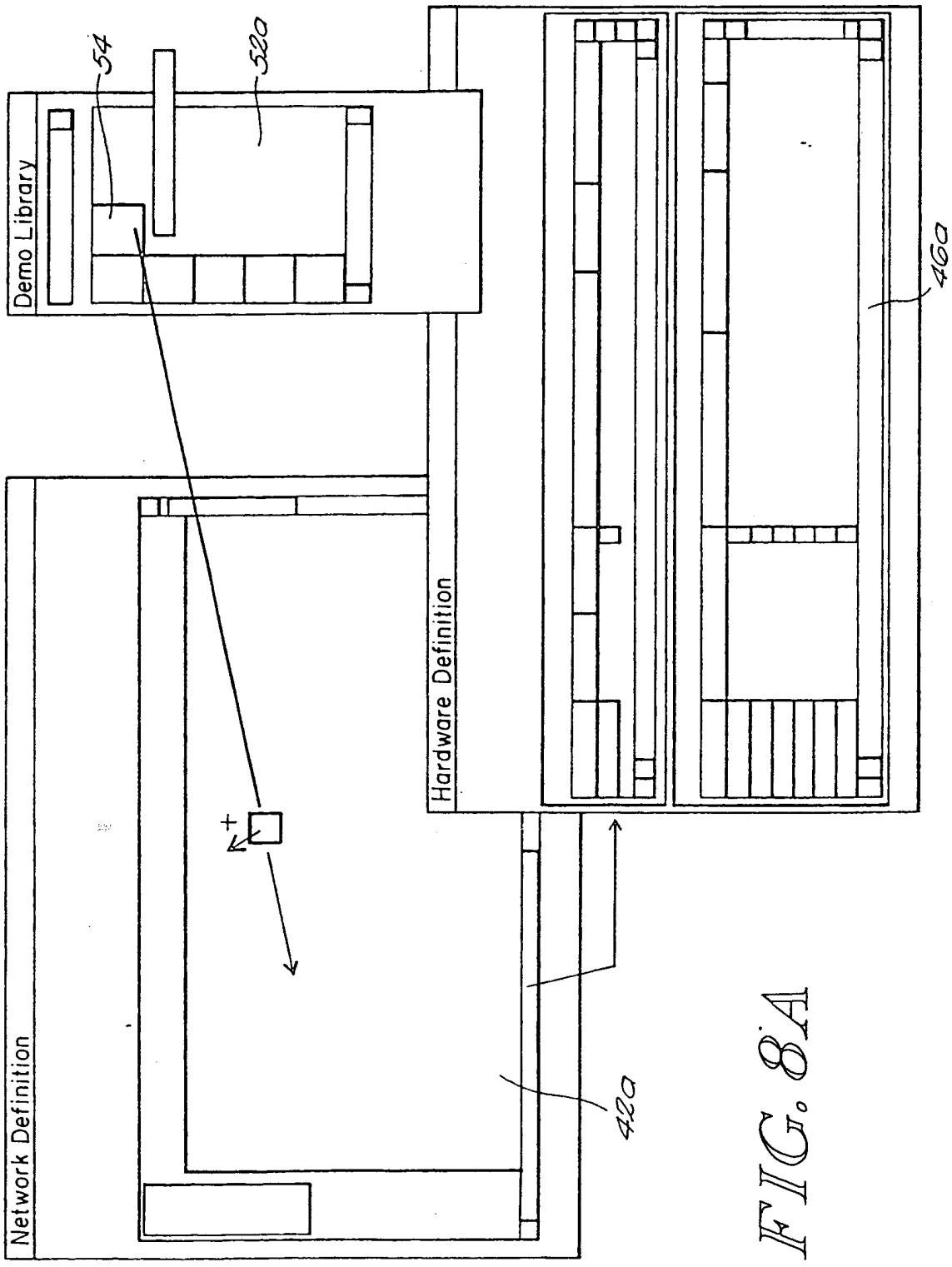


FIG. 8A

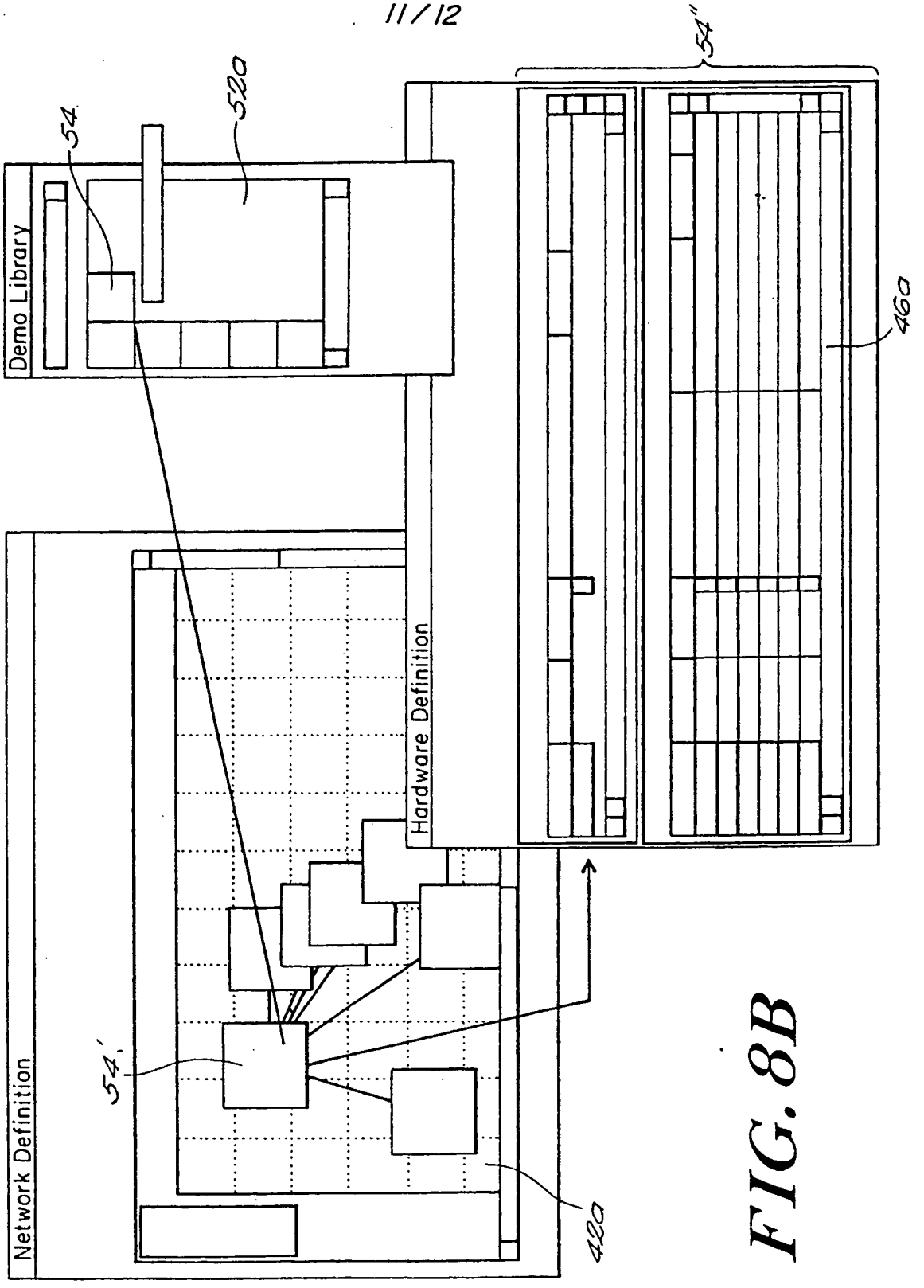
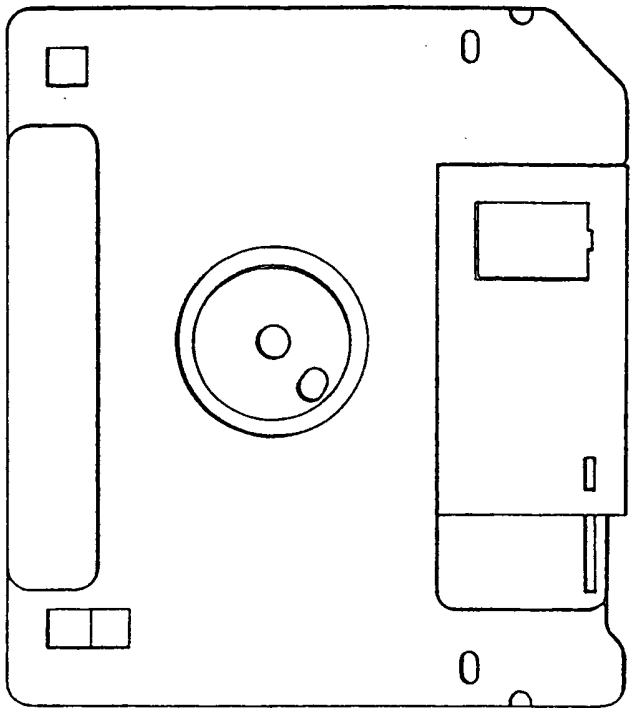

54''
520
54
460
Demo Library
Hardware Definition
Network Definition
54'
420

FIG. 8B



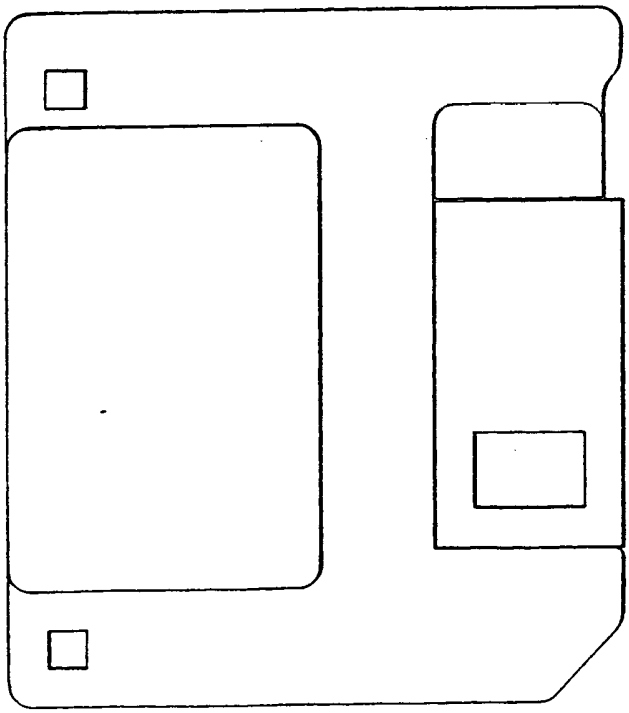


FIG. 9

INTERNATIONAL SEARCH REPORT

National Application No
PCT/US 98/08725

A. CLASSIFICATION OF SUBJECT MATTER
IPC 6 G06F9/46 G06F9/44

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 6 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category '	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 5 315 703 A (MATHENY JOHN R ET AL) 24 May 1994 see column 1, line 31 - line 68 see column 9, line 45 - line 57 see column 11, line 29 - column 12, line 37 see column 17, line 15 - column 18, line 40	1-51
X	WO 94 23365 A (KALIEDA LABS INC) 13 October 1994 see page 1, line 8 - page 6, line 5	1-51
A	EP 0 297 339 A (BMC SOFTWARE INC) 4 January 1989 see column 1, line 1 - column 3, line 10	9,10,16, 36

☐ Further documents are listed in the continuation of box C.

☒ Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
4 August 1998	11/08/1998

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Brandt, J

Form PCT/ISA/210 (second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

Information on patent family members

ıational Application No
PCT/US 98/08725

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 5315703 A	24-05-1994	AU 5984594 A	19-07-1994
		CA 2135527 A	07-07-1994
		DE 69310188 D	28-05-1997
		DE 69310188 T	27-11-1997
		EP 0664026 A	26-07-1995
		JP 8501401 T	13-02-1996
		WO 9415285 A	07-07-1994
		US 5367633 A	22-11-1994
		US 5517606 A	14-05-1996
WO 9423365 A	13-10-1994	US 5430875 A	04-07-1995
		AU 674215 B	12-12-1996
		AU 6419194 A	24-10-1994
		CA 2134684 A	01-10-1994
		EP 0643850 A	22-03-1995
		JP 7508116 T	07-09-1995
EP 0297339 A	04-01-1989	AT 134779 T	15-03-1996
		DE 3855029 D	04-04-1996
		DE 3855029 T	05-09-1996
		US 5566334 A	15-10-1996